

# **Security Quality Requirements Engineering (SQUARE) Methodology**

Nancy R. Mead  
Eric D. Hough  
Theodore R. Stehney II

*November 2005*

TECHNICAL REPORT  
CMU/SEI-2005-TR-009  
ESC-TR-2005-009





**Carnegie Mellon  
Software Engineering Institute**

---

Pittsburgh, PA 15213-3890

# **Security Quality Requirements Engineering (SQUARE) Methodology**

CMU/SEI-2005-TR-009  
ESC-TR-2005-009

Nancy R. Mead  
Eric D. Hough  
Theodore R. Stehney II

*November 2005*

**Networked Systems Survivability Program**

Unlimited distribution subject to the copyright.

This report was prepared for the

SEI Administrative Agent  
ESC/XPK  
5 Eglin Street  
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Christos Sondras  
Chief of Programs, XPK

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2005 Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (<http://www.sei.cmu.edu/publications/pubweb.html>).

# Table of Contents

<b>Acknowledgments .....</b>	<b>vii</b>
<b>Executive Summary.....</b>	<b>ix</b>
<b>Abstract.....</b>	<b>xi</b>
<b>1      Introduction .....</b>	<b>1</b>
1.1     Industry Problem .....	1
1.2     Practice Description.....	2
<b>2      Security Quality Requirements Engineering Process.....</b>	<b>3</b>
<b>3      Security Quality Requirements Engineering Steps .....</b>	<b>7</b>
3.1     Step 1: Agree on Definitions .....	8
3.2     Step 2: Identify Security Goals.....	10
3.3     Step 3: Develop Artifacts.....	12
3.4     Step 4: Perform Risk Assessment.....	13
3.5     Step 5: Select Elicitation Technique .....	14
3.6     Step 6: Elicit Security Requirements.....	15
3.7     Step 7: Categorize Requirements .....	17
3.8     Step 8: Prioritize Requirements .....	18
3.9     Step 9: Requirements Inspection .....	19
<b>4      Recent Results.....</b>	<b>21</b>
4.1     Step 1: Agree on Definitions .....	21
4.2     Step 2: Identify Security Goals.....	23
4.3     Step 3: Develop Artifacts.....	24
4.3.1     System Architecture.....	24
4.3.2     Attack Trees.....	26
4.3.3     Use Cases .....	27
4.3.4     Misuse Cases .....	28
4.3.5     Reconciliation of Attack Trees and Misuse Cases.....	30
4.4     Essential Assets and Services .....	32
4.4.1     Essential Services.....	32
4.4.2     Essential Assets .....	33
4.5     Step 4: Perform Risk Assessment.....	34
4.5.1     Risk Assessment Techniques .....	34
4.5.2     Risk Assessment Field Tests .....	35

4.6	Step 5: Select Elicitation Techniques.....	37
4.6.1	Literature Review .....	37
4.7	Step 6: Elicit Security Requirements .....	39
4.7.1	IBIS .....	39
4.7.2	ARM.....	41
4.7.3	JAD .....	43
4.8	Step 7: Categorize Requirements .....	45
4.9	Step 8: Prioritize Requirements.....	46
4.10	Step 9: Inspect Requirements.....	49
<b>5</b>	<b>Future Research Plans.....</b>	<b>51</b>
<b>Appendix</b>	.....	<b>53</b>
<b>References</b>	.....	<b>59</b>

---

# List of Figures

Figure 1:	Sample of Terms and Definitions Provided to Stakeholders for Review	22
Figure 2:	Simple Hierarchy of Goals and Recommendations .....	23
Figure 3:	Example System Architecture Diagram.....	25
Figure 4:	Example Attack Tree .....	26
Figure 5:	Sample Use Case Diagram.....	28
Figure 6:	Example Misuse Diagram Trace .....	30
Figure 7:	Example IBIS Map Generated with Compendium .....	40
Figure 8:	Cost/Value Diagram of Requirements .....	47



---

# List of Tables

Table 1:	Steps in the SQUARE Process .....	4
Table 2:	Example Set of Initial Terms.....	8
Table 3:	Example Term with Suggested Definitions .....	9
Table 4:	Minimal Set of Categories for Requirements Categorization .....	17
Table 5:	Acme Corporation's Business and Security Goals .....	23
Table 6:	Sample Use Case.....	27
Table 7:	Example Misuse Case .....	28
Table 8:	Mapping Between Misuse Cases and Attack Trees.....	31
Table 9:	Use Cases and Initial Rankings of Essentiality.....	32
Table 10:	Results of Risk Assessment Literature Review .....	35
Table 11:	Risk Assessment Results.....	36
Table 12:	Comparison of Elicitation Techniques.....	38
Table 13:	Requirements Generated for Acme Using the “Unstructured Interview” Elicitation Technique .....	39
Table 14:	Security Requirements Generated with IBIS .....	41
Table 15:	Initial Set of Security Requirements Produced with ARM .....	42
Table 16:	Refined Set of Security Requirements Produced with ARM .....	43
Table 17:	Security Requirements Generated with JAD .....	44
Table 18:	Grouped Security Requirements in ARM.....	45
Table 19:	Pairwise Comparison Matrix in AHP.....	46

---

Table 20: Interpretation of Values in Matrix .....	46
Table 21: Interpretation of Costs in Matrix.....	47
Table 22: Peer Review Log Format.....	49
Table 23: Inspection Checklist .....	50
Table 24: Terms and Definitions from Initial Case Study.....	53
Table 25: Goal Hierarchy for Acme Corporation .....	58

---

## Acknowledgments

The authors would like to acknowledge the work of the student teams at Carnegie Mellon University that have contributed to the SQUARE methodology. They are Peter Chen, Lydia Chung, Marjon Dean, Dan Gordon, Frank Hung, Lilian Lopez, Don Ojoko-Adams, Hassan Osman, Will Salamon, Ted Stehney, Neha Wattas, Nick (Ning) Xie, and Eugene Yu.



---

# Executive Summary

The Software Engineering Institute's Networked Systems Survivability (NSS) Program at Carnegie Mellon University has developed a methodology to help organizations build security into the early stages of the production life cycle. The Security Quality Requirements Engineering (SQUARE) Methodology consists of nine steps that generate a final deliverable of categorized and prioritized security requirements. Although the SQUARE Methodology could likely be generalized to any large-scale design project, it was designed for use with information technology systems.

The SQUARE process involves the interaction of a team of requirements engineers and the stakeholders of an IT project. It begins with the requirements engineering team and project stakeholders agreeing on technical definitions that serve as a baseline for all future communication. Next, business and security goals are outlined. Third, artifacts and documentation are created, which are necessary for a full understanding of the relevant system. A structured risk assessment determines the likelihood and impact of possible threats to the system. Following this work, the requirements engineering team determines the best method for eliciting initial security requirements from stakeholders, which is dependent on several factors, including the stakeholders involved, the expertise of the requirements engineering team, and the size and complexity of the project. Once a method has been established, the participants rely on artifacts and risk assessment results to elicit an initial set of security requirements. Two subsequent stages are spent categorizing and prioritizing these requirements for management's use in making tradeoff decisions. Finally, an inspection stage is included to ensure the consistency and accuracy of the security requirements that have been generated.

SQUARE is a work in progress. Several case studies with real-world clients have shown that the methodology holds good promise for incorporation into industry practice. The SQUARE process has been enhanced and refined throughout the case studies. The current working model is presented in this paper. NSS is currently continuing research on the process and is working in parallel to create a CASE tool to support each stage of the methodology.



---

## **Abstract**

Requirements engineering, a vital component in successful project development, often does not include sufficient attention to security concerns. Studies show that up-front attention to security can save the economy billions of dollars, yet security concerns are often treated as an afterthought to functional requirements. Industry can thus benefit from a model to examine security requirements in the development stages of the production life cycle.

This report presents the Security Quality Requirements Engineering (SQUARE) Methodology for eliciting and prioritizing security requirements in software development projects, which was developed by the Software Engineering Institute's Networked Systems Survivability (NSS) Program. The methodology's steps are explained, and results from its application in recent case studies are examined. The NSS Program continues to develop SQUARE, which has proven effective in helping organizations understand their security posture and produce products with verifiable security requirements.



---

# 1 Introduction

## 1.1 Industry Problem

Requirements engineering is well recognized in industry to be critical to the success of any major development project. Several authoritative studies have shown that requirements engineering defects cost 10 to 200 times more to correct once fielded than they would if they were detected during requirements development [Boehm 88, McConnell 01]. Other studies have shown that reworking requirements defects on most software development projects costs 40 to 50 percent of total project effort [Jones 86], and the percentage of defects originating during requirements engineering is estimated at more than 50 percent [Wiegers 01].

A recent study found that the return on investment when security analysis and secure engineering practices are introduced early in the development cycle ranges from 12 to 21 percent, with the highest rate of return occurring when the analysis is performed during application design [Soo Hoo 01]. NIST reports that software that is faulty in security and reliability costs the economy \$59.5 billion annually in breakdowns and repairs [NIST 02]. The costs of poor security requirements show that there would be a high value to even a small improvement in this area. By the time that an application is fielded and in its operational environment, it is very difficult and expensive to significantly improve its security.

Requirements problems are the single number one reason projects

- exceed budget
- exceed schedule
- have significantly reduced scope
- deliver poor-quality applications
- are not significantly used once delivered
- are cancelled

Requirements engineering typically suffers from the following major problems:

- *Requirements identification* typically does not include all relevant stakeholders and does not use the most modern or efficient techniques.
- *Requirements analysis* typically is either not performed at all (identified requirements are directly specified without any analysis or modeling) or is restricted to functional re-

quirements, ignoring quality requirements and other constraints such as architecture, design, implementation, and testing.

- *Requirements specification* is typically haphazard, with specified requirements that are ambiguous, incomplete (e.g., non-functional requirements are often missing), inconsistent, not cohesive, infeasible, obsolete, unable to be tested or validated, and not usable by all of their intended audiences.
- *Requirements management* is typically limited to tracing, scheduling, and prioritization, with poor storage (e.g., in one or more documents rather than in a database or tool) and missing attributes.

## 1.2 Practice Description

While much has been written about security requirements in the abstract, the mechanisms to translate the theory into practice have been unclear. If security requirements are not effectively defined, the resulting system cannot be effectively evaluated for success or failure prior to implementation. Security requirements are often missing in the requirements elicitation process. The lack of validated methods is considered one of the factors.

In addition to employing applicable software engineering techniques, the organization must understand how to incorporate the techniques into its existing software development processes. The identification of organizational mechanisms that promote or inhibit the adoption of security requirements elicitation can be an indicator of the security level of the resulting product.

An earlier report, focusing on survivable requirements engineering, provided an interesting range of material in this respect [Mead 03]. Security requirements engineering has only been recently researched, with much of the material assembled within the past year or two. The security area did not have techniques or templates for requirements elicitation. By assembling an elicitation framework based on our initial research and applying it to active software development efforts, we were able to identify additional research areas and refine the framework through further research.

If usable approaches to security are developed and mechanisms to promote organizational use are identified, then the organization can ensure that the resulting product effectively meets security requirements.

---

## 2 Security Quality Requirements Engineering Process

Security Quality Requirements Engineering (SQUARE) has been developed at Carnegie Mellon University by Nancy Mead with Donald Firesmith and Carol Woody of the Software Engineering Institute (SEI). The process provides a means for eliciting, categorizing, and prioritizing security requirements for information technology systems and applications. The long-term goal of SQUARE is to integrate security considerations into the early stages of the development life cycle. SQUARE has also proven to be useful for documenting and analyzing the security aspects of fielded systems and has potential for steering future improvements and modifications to these systems.

As is common with many quality requirements, the distinction between *functional* and *non-functional* security requirements of a system is subtle. For instance, a developer may argue that a system *must* protect against denial-of-service attacks in order to fulfill its mission. Such a requirement may be considered by other stakeholders to be an availability or reliability issue that is not central to the mission of the system. In the SQUARE methodology, security requirements are often discussed in the context of quality, non-functional requirements. However, as opposed to artificially categorizing *all* security requirements as nonfunctional, the methodology approaches security requirements as they are often handled in requirements engineering: as an afterthought and add-on to the system's functional requirements. By addressing security requirements in this respect, the SQUARE Methodology is able to accurately adapt to the current state of the practice of software and requirements engineering.

The methodology is most effective and accurate when conducted with a team of requirements engineers with security expertise and the stakeholders of the project. The requirements engineering team can be thought of as external consultants, though often the team is composed of one or more internal developers of the project. Throughout this report, the terms "requirements engineer" and "requirements engineering team" are synonymous. Likewise, this report will refer to "stakeholders" also as "clients" or "the client organization." The effectiveness of SQUARE in eliciting requirements is dependent on representation from the project's stakeholders. Thus, the requirements engineering team must emphasize the importance of establishing a representative set of stakeholders to participate in the methodology.

SQUARE can be decomposed into nine discrete steps, which are outlined in Table 1. Each step identifies the necessary inputs, major participants, suggested techniques, and final output. In Section 3 of this report, the steps are explained in detail. Generally, the output of each step serves as the sequential input to the following steps, though some steps may be per-

formed in parallel. For instance, it might be more efficient for the requirements engineering team to perform Step 2 (Identify Security Goals) and Step 3 (Develop Artifacts) simultaneously, since to some extent they are independent activities. The output of both steps, however, is required for Step 4 (Identify Security Goals).

*Table 1: Steps in the SQUARE Process*

Step Number	Step	Input	Techniques	Participants	Output
1	Agree on definitions	Candidate definitions from IEEE and other standards	Structured interviews, focus group	Stakeholders, requirements team	Agreed-to definitions
2	Identify security goals	Definitions, candidate goals, business drivers, policies and procedures, examples	Facilitated work session, surveys, interviews	Stakeholders, requirements engineer	Goals
3	Develop artifacts to support security requirements definition	Potential artifacts (e.g., scenarios, misuse cases, templates, forms)	Work session	Requirements engineer	Needed artifacts: scenarios, misuse cases, models, templates, forms
4	Perform risk assessment	Misuse cases, scenarios, security goals	Risk assessment method, analysis of anticipated risk against organizational risk tolerance, including threat analysis	Requirements engineer, risk expert, stakeholders	Risk assessment results
5	Select elicitation techniques	Goals, definitions, candidate techniques, expertise of stakeholders, organizational style, culture, level of security needed, cost benefit analysis, etc.	Work session	Requirements engineer	Selected elicitation techniques
6	Elicit security requirements	Artifacts, risk assessment results, selected techniques	Joint Application Development (JAD), interviews, surveys, model-based analysis, checklists, lists of reusable requirements types, document reviews	Stakeholders facilitated by requirements engineer	Initial cut at security requirements

Step Number	Step	Input	Techniques	Participants	Output
7	Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints	Initial requirements, architecture	Work session using a standard set of categories	Requirements engineer, other specialists as needed	Categorized requirements
8	Prioritize requirements	Categorized requirements and risk assessment results	Prioritization methods such as Triage, Win-Win	Stakeholders facilitated by requirements engineer	Prioritized requirements
9	Requirements inspection	Prioritized requirements, candidate formal inspection technique	Inspection method such as Fagan, peer reviews	Inspection team	Initial selected requirements, documentation of decision making process and rationale

As Table 1 illustrates, the first task for the organization is to agree on a common set of security definitions, followed by the definition of organizational security goals. These security goals may be derived from business application goals, potential threats to project assets, and management controls and policy. The requirements engineer can then develop artifacts (network maps and diagrams, attack trees, and use/misuse cases) that will aid in the elicitation of security requirements. Next, a formal risk assessment allows the organization to understand how the likelihood and impact of various threats can affect the project's security goals and assets.

At this point in the SQUARE process, the requirements engineer must select one or more requirements-elicitation techniques appropriate for the organization's culture, expertise, level of security required, and nature of the system being developed. The selected techniques are subsequently used to elicit an initial set of security requirements in the form of operational constraints on the system. An example of such a requirement is "The system shall only reveal employee salary information to members of the Human Resources Department." These requirements are then categorized, prioritized, and formally inspected for correctness in the remaining steps of SQUARE. The final output of the process is a security requirements document that satisfies the security goals of the project, contains clear and verifiable requirements, and is agreed on by all relevant stakeholders.



---

## **3 Security Quality Requirements Engineering Steps**

In this section, the steps in the SQUARE Methodology are enumerated in greater detail. The purpose of this section is to clarify the purpose of each step, the expectations of the participants, and the exit criteria of each step. Recommendations to the process are also stated where appropriate.

### 3.1 Step 1: Agree on Definitions

In order to guarantee effective and clear communication throughout the requirements engineering process, the requirements engineering team and stakeholders must first agree on a common set of terminology and definitions. Given the differences in expertise, knowledge, and experience, an arbitrary term may have multiple meanings between the participants of SQUARE. In addition, there may be ambiguity in the level of detail that is assumed for a given term. For instance, one stakeholder may view “access controls” as a set of policies that governs which users may be granted access to which resources. Another stakeholder may view access controls as the software elements in the system that actually implement this functionality. These differences in perspective must be resolved before the process can continue.

Fortunately, it is not necessary to reproduce a comprehensive set of definitions for each iteration of SQUARE. Using public resources, such as the Software Engineering Body of Knowledge (SWEBOK) [IEEE 05], IEEE 610.12 Standard Glossary of Software Engineering Terminology [IEEE 90], and Wikipedia [Wiki 05], the requirements engineering team can produce and reuse a comprehensive set of security terms with which to work. See Table 2 for an example set of initial terms upon which the requirements engineering team can build.

*Table 2: Example Set of Initial Terms*

access control	corruption	honey pot	non-repudiation	spoof
access control list (ACL)	cracker	impact	patch	SQL injection
antivirus software	denial-of-service (DoS) attack	incident	penetration	stakeholder
artifact	disaster recovery plan	incident handling	penetration testing	stealthing
asset	disclosure	insider threat	physical security	survivability
attack	disgruntled employee	integrity	port scanning	target
audit	downtime	interception	privacy	threat
authentication	disruption	interruption	procedure	threat assessment
availability	encryption	intrusion	recognition	threat model
back door	espionage	intrusion detection system (IDS)	recovery	toolkits
breach	essential services	liability	replay attack	Trojan
brute force	exposure	luring attack	resilience	trust
buffer overflow	fabrication	malware	resistance	uptime
cache cramming	fault line attacks	man-in-the-middle attack	risk	victim
cache poisoning	fault tolerance	masquerade	risk assessment	virus
confidentiality	firewall	modification	security policy	vulnerability
control	hacker	non-essential services	script kiddies	worm

The initial list of terms should also include suggested definitions for each term and their corresponding sources. This allows the stakeholders to get a general understanding and scope of each term, and in the common case, select one of the suggested definitions as final. See Table 3 for an example of the information that should be provided with each term. In this example, the stakeholders could place a checkmark next to the definition that suits them best.

*Table 3: Example Term with Suggested Definitions*

<b>confidentiality</b>	<input type="checkbox"/> The property that information is not made available or disclosed to unauthorized individuals, entities, or processes. (i.e., to any unauthorized system entity)	[SANS 03a]
	<input type="checkbox"/> Ensuring that information is available to only those with authorized access.	[ISO 04]
	<input type="checkbox"/> Restricting access to information via a hierarchy of classes of access.	[JONES 02]
	<input type="checkbox"/> Other: _____	

### **Requirements Engineering Team Responsibilities:**

1. Provide an initial set of terms with corresponding suggested definitions. All external sources should be cited. The set of terms should be as comprehensive as possible, even if some terms appear to be irrelevant to the project.
2. Provide a means for stakeholders to review and select a desired definition for each term. This process could take place by way of a Web-based tool, email exchanges, or paper surveys. The chosen means must allow the stakeholders to freely add new terms and definitions to the set.
3. Document and share the finalized set of terms and definitions.

### **Stakeholder Responsibilities:**

Select an existing or create a custom definition for each term provided by the requirements engineering team. All stakeholders must come to a consensus on each term's definition in a timely manner and present their results to the requirements engineering team.

### **Joint Responsibilities:**

Establish a single point of contact (POC) for interaction between the requirements engineering team and the stakeholders.

### **Exit Criteria:**

A well-documented, agreed-on set of definitions has been established and is available to all stakeholders and the requirements engineering team. The definitions document will be used as a reference throughout the rest of the SQUARE process.

## **3.2 Step 2: Identify Security Goals**

The purpose of Step 2 in SQUARE is for the stakeholders to formally agree on a set of prioritized security goals for the project. Without overall security goals for the project, it is impossible to identify the priority and relevance of any security requirements that are generated. In addition, the establishment of security goals scopes the rest of the SQUARE process.

Initially, different stakeholders will likely have different security goals. For example, a member of human resources may be concerned about maintaining the confidentiality of personnel records, whereas a stakeholder in finance may be concerned with ensuring that financial data is not modified without authorization. The security goals of the stakeholders may also conflict with one another. A security-conscious stakeholder may place high importance on strong security controls for the system, which in turn may hamper overall system performance. Decreased performance might likely be at odds with the goals of the marketing department. Step 2 in the SQUARE process serves to eliminate such conflicts and align all of the stakeholders' interests.

The security goals of the project must be in clear support of the project's overall business goal, which also must be identified and enumerated in this step. On average, stakeholders should attempt to brainstorm to come up with approximately half a dozen security goals for the project, with more or less depending on the scale of the project. More sophisticated techniques for mapping high-level business requirements to low-level requirements can be found in *Core Security Requirements Artefacts* [Moffett 04] and “Mapping Mission-Level Availability Requirements to System Architectures and Policy Abstractions” [Watro 01].

Once the goals of the various stakeholders have been identified, they must be prioritized. In the absence of consensus, an executive decision may be needed to prioritize the goals.

Finally, the requirements engineering team must encourage the stakeholders to generate security goals as opposed to requirements or recommendations. There is a fine line between a *security goal* such as “The system shall be available for use when needed,” a *requirement* such as “The system must have a continuity of operations plan in place to ensure appropriate system availability,” and a *recommendation* such as “Invest in backup information technology hardware to ensure business continuity.” The requirements engineering team must act as the experts in this situation, providing assistance to the stakeholders so that they may generate an appropriately scoped set of security requirements.

### **Requirements Engineering Team Responsibilities:**

1. Facilitate the brainstorm session by the stakeholders, emphasizing the importance of creating a single business goal, followed by several security goals that support it.
2. Review the stakeholders' business and security goals, providing any feedback on scope, level of detail, and relevance to the business goal of the project.
3. Document and share the finalized business goal and corresponding security goals.

**Stakeholder Responsibilities:**

1. Identify a single business goal for the project. This goal should be stated in one sentence, such as “The system shall provide the means to effectively manage company resources in a disaster situation.”
2. Brainstorm and create approximately half a dozen security goals that are in clear support of the business goal. For example, “The system shall maintain high availability, even in the face of public utility failures.”
3. Prioritize the security goals.
4. Provide the business goal and security goals to the requirements engineering team for review, and edit the goals as deemed necessary by the team.

**Exit Criteria:**

A single business goal for the project and several prioritized security goals that support it have been established.

### **3.3 Step 3: Develop Artifacts**

Before the requirements engineering team and stakeholders can generate a comprehensive set of security requirements, the team must collect a complete set of artifacts of the system. The following are the types of artifacts that should be collected:

- system architecture diagrams
- use case scenarios/diagrams
- misuse case scenarios/diagrams
- attack trees
- standardized templates and forms

In developing such artifacts, it is important to enlist the assistance of knowledgeable engineers from the organization. In some cases, it is possible that the client organization will not have *any* of the artifacts in place, including such basic items as system architecture diagrams. In such situations, the requirements engineering team should reiterate to the stakeholders that by creating and documenting artifacts of the system, they are investing in the success of the project.

#### **Requirements Engineering Team Responsibilities:**

Work with the stakeholders and client organization to identify and collect as many artifacts as possible.

#### **Stakeholder Responsibilities:**

Generate or collect any system artifacts and present them to the requirements engineering team. This information includes system architectures, actors, use/misuse cases, and suspected attacks. It is possible that other experts from within the organization may be called upon to provide information as well.

#### **Joint Responsibilities:**

Verify the accuracy and completeness of all artifacts.

#### **Exit Criteria:**

A set of artifacts for the system, as complete as possible, has been generated by the requirements engineering team and shared with the stakeholders.

## **3.4 Step 4: Perform Risk Assessment**

The purpose of this step in the SQUARE process is to identify the vulnerabilities and threats that face the system, the likelihood that the threats will materialize as real attacks, and any potential consequences of an attack. Without a risk assessment, organizations can be tempted to implement security requirements or countermeasures without a logical rationale. For instance, the stakeholders may decide that encryption is a necessary component of their system without fully understanding the nature of the problem that encryption can solve. The risk assessment also serves to prioritize the security requirements at a later stage in the process.

There are a growing number of risk assessment methods from which to choose (see the list of examples in Section 4.5.1). Some of the methods are very structured and may require the assistance of an external risk expert. Ideally, this expert would already be a part of the requirements engineering team.

After the threats have been identified by the risk assessment method, they must be classified according to likelihoods. Again, this will aid in prioritizing the security requirements that are generated at a later stage. For each threat identified, a corresponding security requirement can identify a quantifiable, verifiable response. For instance, a requirement may describe speed of containment, cost of recovery, or limit to the damage that can be done to the system's functionality.

### **Requirements Engineering Team Responsibilities:**

1. Facilitate the completion of a structured risk assessment, likely performed by an external risk expert.
2. Review the results of the risk assessment and share them with stakeholders.

### **Exit Criteria:**

All vulnerabilities and threats have been identified and classified according to their likelihoods. Potential consequences of attacks are identified. The results are well documented and shared with the stakeholders.

### **3.5 Step 5: Select Elicitation Technique**

The requirements engineering team must select an elicitation technique that is suitable for the client organization and project. Although this task may appear to be straightforward, it is often the case that multiple techniques will likely work for the same project. The difficulty is in choosing a technique that can adapt to the number and expertise of stakeholders, size and scope of the client project, and expertise of the requirements engineering team. It is extremely unlikely that any single technique will work for all projects under all circumstances, though previous experience has shown that the Accelerated Requirements Method (ARM) has been successful in eliciting security requirements. This particular technique is discussed further in Section 4.7.2.

The following is a sample of elicitation techniques that may be appropriate:

- Structured/unstructured interviews
- Use/misuse cases [Jacobson 92]
- Facilitated meeting sessions, such as Joint Application Development and the Accelerated Requirements Method [Wood 89, Hubbard 99]
- Soft Systems Methodology [Checkland 89]
- Issue-Based Information Systems [Kunz 70]
- Quality Function Deployment [QFD 05]
- Feature-Oriented Domain Analysis [Kang 90]
- Controlled Requirements Expression [Mullery 79]
- Critical Discourse Analysis [Schiffrin 94]

#### **Requirements Engineering Team Responsibilities:**

1. Select an elicitation technique that is appropriate for the number and expertise of stakeholders, size and scope of the project, and expertise of the requirements engineering team.
2. Document the rationale for the choice and make necessary preparations to execute the technique.

#### **Exit Criteria:**

The requirements engineering team has selected an appropriate elicitation technique and documented the rationale for their choice.

## **3.6 Step 6: Elicit Security Requirements**

This step is the heart of the SQUARE process: the elicitation of security requirements. To the benefit of the requirements engineering team, most elicitation techniques provide detailed guidance on how to perform the elicitation, so this step is simply a matter of executing the technique. However, even if the stakeholders are very knowledgeable about the project and communicate effectively, it can be challenging for the requirements engineering team to elicit correct requirements.

Perhaps the largest mistake that the requirements engineering team can make in this step is to elicit non-verifiable or vague, ambiguous requirements. Each requirement must be stated in a manner that will allow relatively easy verification once the project has been implemented. For instance, the requirement “The system shall improve the availability of the existing customer service center” is impossible to measure objectively. Instead, the requirements engineering team should encourage the production of requirements that are clearly verifiable and, where appropriate, quantifiable. A better version of the previously stated requirement would thus be “The system shall handle at least 300 simultaneous connections to the customer service center.”

A second mistake that the requirements engineering team can make in this step is to elicit *implementations* or *architectural constraints* instead of requirements. Requirements are concerned with *what* the system should do, not *how* it should be done.

All elicitation techniques will involve face-to-face interaction with the stakeholders, so it is also the responsibility of the requirements engineering team to make logistical arrangements with the stakeholders and inform them of the time they can expect to spend in this part of the SQUARE process.

### **Requirements Engineering Team Responsibilities:**

1. Execute the elicitation technique chosen in Step 5. This may entail a large amount of logistical preparation and orientation for the stakeholders. Stakeholders should be informed of the amount of time they can be expected to spend during this step of the process.
2. Document the requirements as they are collected.

### **Stakeholder Responsibilities:**

Follow the instructions given by the requirements engineering team during the elicitation process.

### **Joint Responsibilities:**

Encourage the generation of verifiable, preferably quantifiable security requirements.

**Exit Criteria:**

An initial set of security requirements for the system has been elicited and documented. It is not necessary that the set be considered final or completely correct.

## 3.7 Step 7: Categorize Requirements

The purpose of this step is to allow the requirements engineer and stakeholders to classify the requirements as essential, non-essential, system level, software level, or as architectural constraints. The requirements engineering team can provide to the stakeholders a matrix such as the one in Table 4 to assist in this process.

*Table 4: Minimal Set of Categories for Requirements Categorization*

	System level	Software level	Architectural constraint
Essential			
Non-essential			

The categories in Table 4 are not fixed; each iteration of SQUARE will likely produce a much larger set of categories that are customized to the project at hand. These categories are instead suggested as a minimal set.

Since the goal of SQUARE is to produce security requirements, the requirements engineering team and stakeholders should avoid producing architectural constraints. Architectural constraints are provided as a category here to serve as an outlet for “requirements” that, upon categorization, are considered to be constraints. Ideally, such anomalies would be identified and corrected in the previous steps of the process.

Once the requirements are categorized, the requirements engineering team and stakeholders will be able to prioritize them more efficiently.

### **Requirements Engineering Team Responsibilities:**

1. Provide a baseline set of categories such as those in Table 4. The team may have to suggest alternative categories, depending on the client project.
2. Facilitate the stakeholders’ categorization process.

### **Stakeholder Responsibilities:**

Come to a consensus on the categorization for each requirement.

### **Exit Criteria:**

The initial set of requirements has been organized into stakeholder-defined categories, and any remaining architectural constraints are identified as such.

## **3.8 Step 8: Prioritize Requirements**

In most cases, the client organization will be unable to implement *all* of the security requirements due to lack of time, resources, or developing changes in the goals of the project. Thus, the purpose of this step in the SQUARE process is to prioritize the security requirements so that the stakeholders can choose which requirements to implement and in what order. The results of Step 4, the risk assessment, and Step 7, categorization, are crucial inputs to this step.

The available prioritization methods are flexible and can be as simple as unstructured deliberation between the stakeholders. There are several structured prioritization techniques that exist, such as Triage [Davis 03], Win-Win [Boehm 01], and the Analytical Hierarchy Process (AHP); the latter has been reported to be quite effective [Karlsson 97, Saaty 80]. AHP is discussed in detail in Section 4.9 of this report. Ideally, the requirements engineering team should also produce a cost-benefit analysis to aid the stakeholders' decisions.

During prioritization, some of the requirements may be deemed to be entirely unfeasible to implement. In such cases, the requirements engineering team has a choice: completely dismiss the requirement from further consideration, or document the requirement as “future work” and remove it from the working set of project requirements. This decision should be made after consulting with the stakeholders.

### **Requirements Engineering Team Responsibilities:**

Facilitate the prioritization process with the stakeholders. If a structured prioritization process is selected, teach the stakeholders how to perform the process.

### **Stakeholder Responsibilities:**

Prioritize the security requirements using the risk assessment and categorization results as a basis for decision making.

### **Exit Criteria:**

All security requirements have been prioritized.

## **3.9 Step 9: Requirements Inspection**

The last step of the SQUARE process, requirements inspection, is one of the most important elements in creating a set of accurate and verifiable security requirements. Inspection can be done at varying levels of formality, from Fagan Inspections to peer reviews [Fagan 86, Wiegers 02]. The goal of any inspection method, however, is to find any defects in the requirements such as ambiguities, inconsistencies, or mistaken assumptions.

### **Requirements Engineering Team Responsibilities:**

Facilitate the inspection process by providing any orientation to the structured inspection technique or informal inspection guides such as checklists.

### **Stakeholder Responsibilities:**

Come to a consensus on the validity of each security requirement. Verify that each requirement is verifiable, in scope, within financial means, and feasible to implement. Requirements that do not fit these criteria should have been identified in earlier stages of SQUARE, but the stakeholders should use this opportunity as a last chance to remove any requirements from the working set.

### **Joint Responsibilities:**

Verify that each requirement is directly applicable to one or more of the security goals of the project or in support of a higher level requirement.

### **Exit Criteria:**

All security requirements have been verified both by the requirements engineering team and the stakeholders. At this point the SQUARE process is complete, and the requirements engineering team can produce the final security requirements document for the stakeholders.



---

## 4 Recent Results

The SQUARE Methodology has undergone several case studies conducted by graduate students at Carnegie Mellon University [Chen 04, Gordon 05]. The goals of the case studies were to experiment with each step of the SQUARE process, make recommendations, and determine the feasibility of integrating SQUARE into standardized software development practices. The case studies involved real-world clients that were developing large-scale IT projects. The clients included an IT firm in Pittsburgh, Pennsylvania, a federal government research institute, and a department of the federal government. In this section, the results from the case studies are presented at each step of the process.

### 4.1 Step 1: Agree on Definitions

The process of establishing and agreeing on definitions was conducted rather straightforwardly with the stakeholders in the case studies. The requirements engineering team initially held a brainstorming session to create the preliminary list of terms involved with the security industry. This list was primarily based on the team's academic courses and prior work experience. Table 2 outlines some of these terms, and Section 0 lists the final set of terms along with their agreed-on definitions.

After narrowing the potential definitions for each term down to two or three, along with a "suggested" definition, the team sought input from the stakeholders by emailing a set of compiled definitions with appropriate referencing. The team asked the client to indicate which definitions best fit their understanding, to modify or create new definitions, and to add any terms that may have been omitted. The following figure shows a snapshot of part of the document that was submitted:

Control:

- a. Procedures, which can reduce, or eliminate, the risk of a threat becoming an incident. [1]
- b. An action, device, procedure or technique that removes or reduces vulnerability. [2]  
[recommended]
- c. \_\_\_\_\_  
\_\_\_\_\_

Corruption:

- a. A threat action that undesirably alters system operation by adversely modifying system functions or data. [1] [recommended]
- b. \_\_\_\_\_  
\_\_\_\_\_

Cracker:

- a. Someone who breaks into someone else's computer system, often on a network; bypasses passwords or licenses in computer programs; or in other ways intentionally breaches computer security. [13] [recommended]
- b. \_\_\_\_\_  
\_\_\_\_\_

Denial-of-Service (DoS) Attack:

- a. One in which a multitude of compromised systems attack a single target, thereby causing denial of service for users of the targeted system. The flood of incoming messages to the target system essentially forces it to shut down, thereby denying service to the system to legitimate users. [14]
- b. A Dos Attack is a form of attacking another computer or company by sending millions of more requests every second causing the network to slow down, cause errors or shut down. [15] [recommended]
- c. \_\_\_\_\_  
\_\_\_\_\_

*Figure 1: Sample of Terms and Definitions Provided to Stakeholders for Review*

The clients relied on the requirements engineering team's expertise and knowledge for guidance through the first step; stakeholders considered some terms to overlap and have ambiguous or double meanings. For a large portion of the terms, the suggested definition was selected by the stakeholders. After each term had a finalized definition, the requirements engineering team finalized the dictionary of terms and shared it with the stakeholders.

## 4.2 Step 2: Identify Security Goals

The identification of security goals for each of the case studies was performed with varied difficulty. Some of the case studies had identified business and security goals before the SQUARE process even began. Others did not even have an explicit business goal for their system. In this section of the report, the development of Acme Corporation's security goals is presented as an example.

To begin with, the student team asked the stakeholders of Acme to develop a business goal that could be expressed in a single sentence. In this case, the stakeholders came up with "The system allows the client to make informed decisions based on which assets are available." The team then asked the stakeholders to generate a few security goals for the product that were in more or less clear support of this business goal. They created a simple hierarchy, shown in Figure 2, that illustrated how the system's business goal, security goals, security requirements, and recommendations connected together. In this case, Acme developed one business goal and three security goals, which are shown in Table 5. The corresponding security requirements, which were generated in a later stage of the process, are presented in Section 4.7.



Figure 2: Simple Hierarchy of Goals and Recommendations

Table 5: Acme Corporation's Business and Security Goals

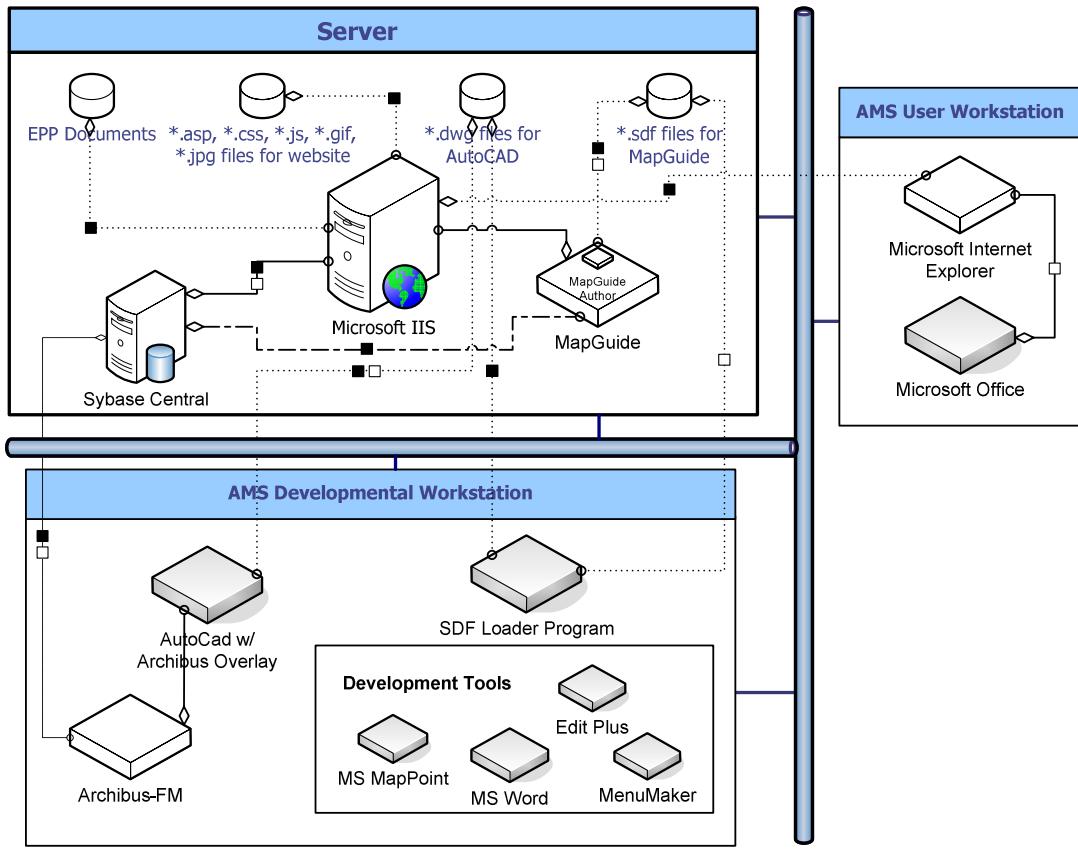
Business Goal	
The system allows the client to make informed decisions based on which assets are available.	
Security Goals	
G-01 Management shall exercise effective control over the system's configuration and usage.	
G-02 The confidentiality, accuracy, and integrity of the system's data shall be maintained.	
G-03 The system shall be available for use when needed.	

## **4.3 Step 3: Develop Artifacts**

Throughout the case studies, a series of artifacts were developed that served as important inputs to subsequent steps. The artifacts included system architecture diagrams, use/misuse cases, attack trees, and assessment of essential assets and services.

### **4.3.1 System Architecture**

The case studies revealed that some organizations, surprisingly, had no documentation of the system architecture for their projects. In some cases, this was due to the dynamic nature of the project: there was no “typical” setup of the system due to the fact that the end user more or less defines that. Regardless, the establishment of system architecture diagrams proved very useful throughout the later stages of SQUARE. It is possible that the requirements engineering team will need to handle very detailed system architecture diagrams. The following figure is an example of a simple system architecture diagram that was produced by one of the case studies.



#### LEGEND

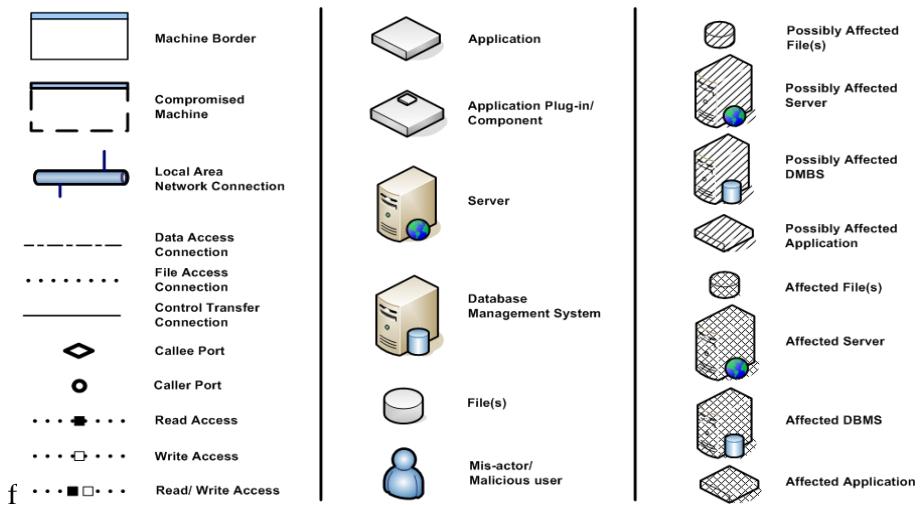


Figure 3: Example System Architecture Diagram

### 4.3.2 Attack Trees

Attack trees provide a formal, hierarchical way of describing the security threats to a system based on the types of attacks that could happen and how they could be realized. Attack tree diagrams represent attacks in a tree structure, where an attacker's goal is listed as the root node and tree leaves represent different ways to achieve that goal. In the following example, a higher level attack tree is depicted, followed by a drill down of a more specific sub tree:

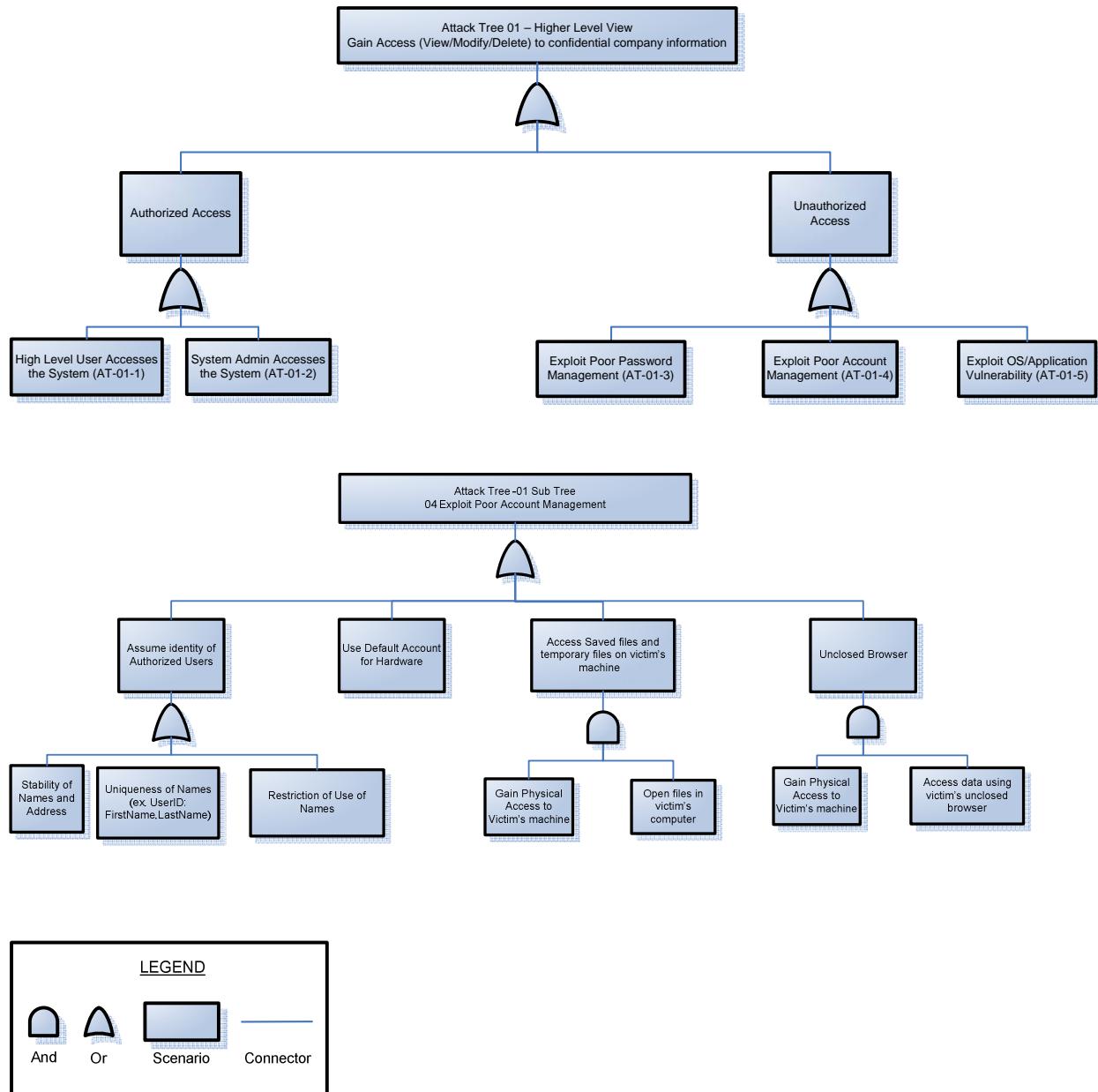


Figure 4: Example Attack Tree

In general, attack trees proved very useful to both the stakeholders and the requirements engineering team in getting a better sense of the scope of threats that the system faces. Many attack trees can be reused between iterations of SQUARE, thus saving the requirements engineering team from developing entirely new attack trees for the same attacks.

### 4.3.3 Use Cases

Use cases are scenario-based artifacts that force stakeholders to answer “how is this done” questions from a user’s perspective. By providing a context for operation, stakeholders and the requirements engineering team can gain a deep understanding of the interactions of the system components. Use case scenarios can be supplemented with use case diagrams, which graphically display the component interactions of the system.

A detailed set of use cases were completed during the SQUARE case studies, which both the requirements engineering team and stakeholders found to be very useful. A sample use case and diagram are shown in Table 6 and Figure 5:

*Table 6: Sample Use Case*

Number	UC-01
Use Case	View Floor Plans
Description	All level of users able to access the system will have the ability to view authorized system information per the Access Control List such as floor plans, damaged areas, employee locator, etc.
Actors	Low-Level User, Medium-Level User, High-Level User, or System Administrator
Assumptions	System Admin has added viewing privileges to the Access Control List System is available Data entered is correct
Steps	User will enter the URL associated with the system  User will receive a prompt to log in their user name and password.  The system authorizes and authenticates the user, then allowed into the system.  The system will allow them to access privileges as specified by the Access Control List.  From here, the user will navigate to Operations/ Maintenance. The user can choose appropriate property and then floor plans.
Variations	Once logged in, the user can also click on the floor plans tab on the right hand side of the system’s main page.
Non-Functional	They will not have edit privileges; view-only privileges will be assigned. If the user attempts to access unauthorized information, the system will display a pop-up window stating that the user is not authorized to access this information.

Related Misuse Cases	MC-01, MC-08, MC-11, MC-12, MC-13, MC-14, MC-15, MC-16, MC-17, MC-18, MC-19, MC-20, MC-21, MC-22
----------------------	--

### UC-01: View Floor Plans

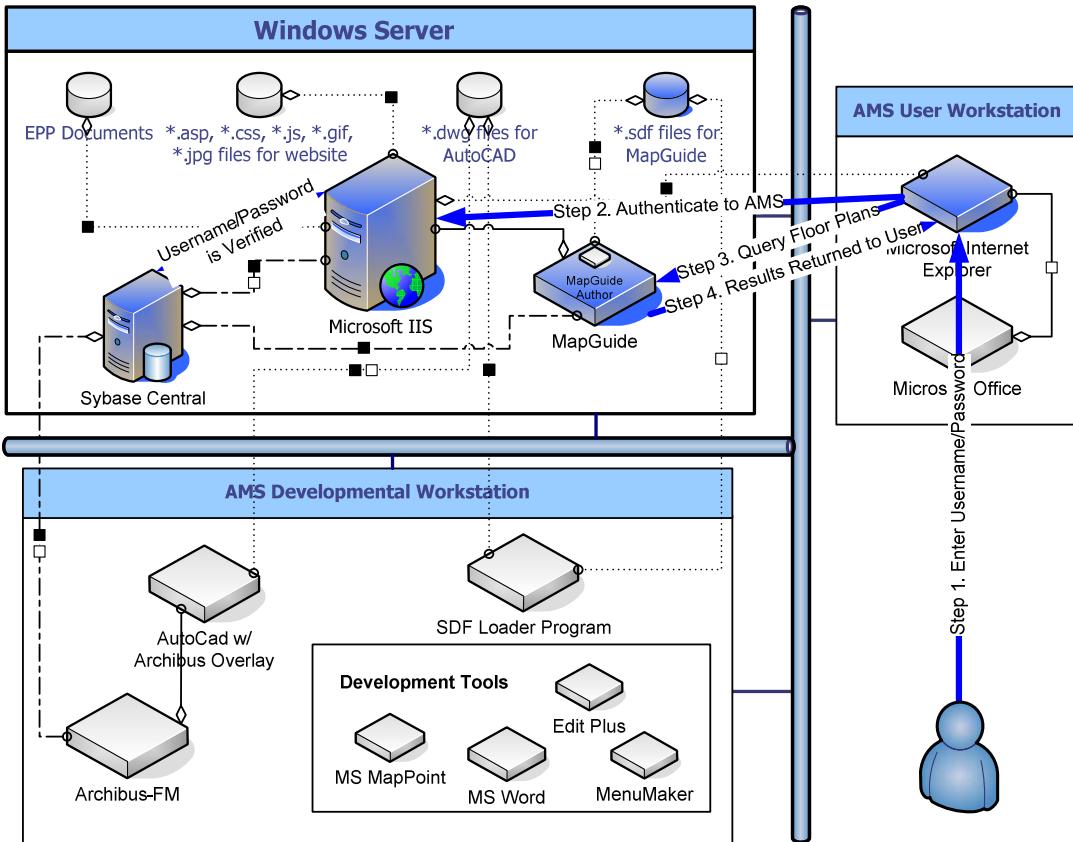


Figure 5: Sample Use Case Diagram

#### 4.3.4 Misuse Cases

For the initial case study, a detailed set of misuse cases were completed that encompassed the most significant threats to the system. Sample misuse case and diagram traces are shown in Table 7 and Figure 6.

Table 7: Example Misuse Case

<b>Number:</b>	MC-01
<b>Name:</b>	Unauthorized logon to server.
<b>Scope:</b>	User Authorization Concerns
<b>Priority:</b>	<input type="checkbox"/> Low <input type="checkbox"/> Medium <input checked="" type="checkbox"/> High
<b>Deployment Environment:</b>	<input checked="" type="checkbox"/> Intranet <input type="checkbox"/> Extranet/Internet

<b>Mis-actors:</b>	Unauthorized users								
<b>Access Right Levels:</b>	<input checked="" type="checkbox"/> Low-Level System Users <input checked="" type="checkbox"/> Medium-Level System Users <input checked="" type="checkbox"/> High-Level System Users <input checked="" type="checkbox"/> Sys Admin <input type="checkbox"/> Other Network User								
<b>Point of Entry:</b>	<input type="checkbox"/> Network <input checked="" type="checkbox"/> Host <input type="checkbox"/> Application								
<b>Security Attributes Affected:</b>	<input checked="" type="checkbox"/> Confidentiality <input checked="" type="checkbox"/> Integrity <input type="checkbox"/> Availability								
<b>Description:</b>	An unauthorized user attempts to log on to the server and succeeds.								
<b>Sophistication:</b>	<input checked="" type="checkbox"/> Low <input type="checkbox"/> Medium <input type="checkbox"/> High								
<b>Pre-conditions:</b>	Access control lists are configured properly in a domain based network. The unauthorized user has unintended logon rights to the Server. The Server resides on an intranet network								
<b>Assumptions:</b>	The user does not have expressed permission to log on to the server.								
<b>Post-conditions:</b>	<table border="1"> <tr> <td>Worst Case Threat:</td> <td>The unauthorized user logs onto the server machine. Her actions are never caught.</td> </tr> <tr> <td>Wanted Prevention Guarantee:</td> <td>Enforce machine access control list (ACL) security policy. (role-based user authentication)</td> </tr> <tr> <td>Wanted Detection Guarantee:</td> <td>Logon attempts are logged and viewed by system administrators.</td> </tr> <tr> <td>Wanted Recovery Guarantee:</td> <td>Remove users' unauthorized logon rights on the server.</td> </tr> </table>	Worst Case Threat:	The unauthorized user logs onto the server machine. Her actions are never caught.	Wanted Prevention Guarantee:	Enforce machine access control list (ACL) security policy. (role-based user authentication)	Wanted Detection Guarantee:	Logon attempts are logged and viewed by system administrators.	Wanted Recovery Guarantee:	Remove users' unauthorized logon rights on the server.
Worst Case Threat:	The unauthorized user logs onto the server machine. Her actions are never caught.								
Wanted Prevention Guarantee:	Enforce machine access control list (ACL) security policy. (role-based user authentication)								
Wanted Detection Guarantee:	Logon attempts are logged and viewed by system administrators.								
Wanted Recovery Guarantee:	Remove users' unauthorized logon rights on the server.								
<b>Potential Mis-actor Profiles:</b>	Medium to highly skilled, potentially host administrators with medium criminal intent.								
<b>Stakeholders and Threats:</b>	Stakeholders' clients: loss of data integrity and/or confidentiality. Stakeholders: loss of reputation, loss of current and potential clients.								
<b>Related Use Cases:</b>	UC-01, UC-02, UC-03, UC-04, UC-05, UC-06, UC-07, UC-08								
<b>Related Threats:</b>	Elevation of privilege, disclosure of confidential data, unauthorized access to administration interface, unauthorized access to configuration stores, retrieval of print text configuration secrets								
<b>Architectural Recommendation:</b>	(AR-01) All shared drives on the network should enforce authentication policies. (AR-03) Audit information is stored in a separate location from the servers and the workstations. (AR-19) Implement role-based authentication control.								

<b>Policy Recommendation:</b>	(PR-03) Audit information must be reviewed routinely. (monthly) (PR-04) Applications and operating systems must be patched routinely. (bi-monthly) (PR-07) Enforce strong password policies. (PR-13) Password protects any necessary shared documents. (PR-16) Require users to change their passwords periodically. (monthly) (PR-19) Set clear and defined user access controls for all users. (Low, Medium, High, System Admins). (PR-20) Perform routine system and data back-up. (weekly) (PR-21) User activities must be periodically reviewed. (bimonthly) (PR-23) Users should not have rights or access levels beyond those of which prescribed by his or her job responsibilities. (PR-24) Users should not reveal their account names and passwords in any given situations.
-------------------------------	--

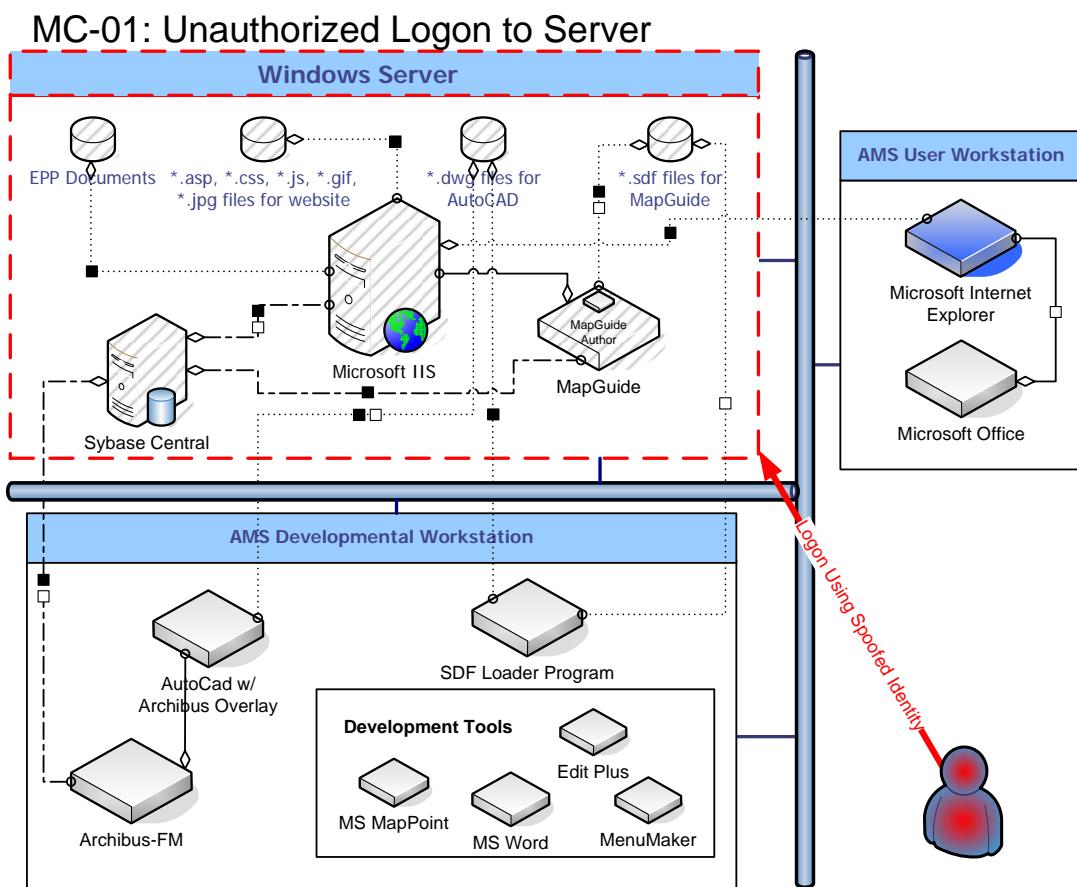


Figure 6: Example Misuse Diagram Trace

#### 4.3.5 Reconciliation of Attack Trees and Misuse Cases

One of the goals of completing the attack trees was to ensure that the team had a complete set of misuse cases. Table 8 shows a mapping between the attack trees and the misuse cases.

While the attack trees provide a general picture of potential attacks on the system, the misuse

cases drill down to the details of the interactions between system components in the event of an attack.

**Table 8: Mapping Between Misuse Cases and Attack Trees**

Misuse Case Name	Attack Tree
Unauthorized logon on the Windows 2003 server	AT-01-04
Sys Admin gain access to system data	AT-01-02
Users gain Sys Admin rights on the Windows 2003 server (Elevation of Privilege)	AT-01-04
Sys Admin deletes critical system configurations on the Windows 2003 server	AT-01-02
Sys Admin creates holes in the system configurations on the Windows 2003 server	AT-01-02
User deletes critical data from the AMS system	AT-01-03
Users falsify system data	AT-01-03
Access system data through developmental machines	AT-01-01, 02
Access system data directly to/from database	AT-01-01, 02
Steal user credential information through developmental machines	AT-01-01, 02
Users see data that they should not see from their workstations	AT-01-01, 02, 03
Malicious user uses replay attack in the same browser to assume the identity of another user	AT-01-05
Malicious users tap communications channel between workstations and servers	AT-01-05
Malicious users gain access to sensitive data via saved Excel export files on victim's machine	AT-01-05
Malicious users install malicious programs that can tap into Excel's memory to steal exported data	AT-01-05
Input validation attack	AT-01-05
Infect Windows 2003 server with virus/worms	AT-01-05
User gains access to the system using spoofed identities	AT-01-04
Information gathering/network eavesdropping	AT-01-05
Brute force attacks: password cracking/credential theft	AT-01-03
Denial of service	AT-02-01
Execute malicious code	AT-01-05

Mapping the attack trees to the misuse cases provided a useful sanity check for the work. Additionally, the team found that it might have been useful to have two independent teams working in parallel: one team working on attack trees and one team working on misuse cases.

## 4.4 Essential Assets and Services

In one of the case studies, the team noticed that another class of artifacts could be derived by utilizing the Survivable Systems Analysis (SSA) method developed by the CERT® Coordination Center (CERT/CC) [CERT/CC 02]. SSA is a white-team exercise aimed at providing survivable recommendations for a system. Some of the preliminary work for SSA is already covered in the earlier stages of SQUARE. The team noticed that Step 2 of SSA, defining essential service scenarios and components, would yield useful artifacts for inclusion in the case study.

To begin identifying the essential elements of the system, the team first looked back to the business goal of the client's project. In this case, the goal was to "provide the ability to make important decisions in emergency situations based upon current and available information." The students analyzed the use cases of the system and made a determination as to which services, assets, and components were essential to fulfilling the business goal of Acme's product.

### 4.4.1 Essential Services

The team analyzed the importance of each of the major system services, outlined in Table 9 by way of use cases, and made a determination as to each service's essentiality.

*Table 9: Use Cases and Initial Rankings of Essentiality*

Use Case	Service	Status
UC-1	View floor plans	Essential
UC-2	Enter damage assessment	Essential
UC-3	Add/delete/edit Post-It notes	Non-Essential
UC-4	Find specialized employees	Important
UC-5	Create journal entry	Non-Essential
UC-6	Install the base system software	Non-Essential
UC-7	Create links to documents	Non-Essential
UC-8	Archibus admin: add user and assign privileges	Non-Essential
UC-9	View contact information for maintenance tasks	Important
UC-10	Create open space report	Essential
UC-11	View incident command	Essential

The major business goal of this particular system was to allow decisions to be made both before an emergency takes place (i.e., in the planning phase), as well as during and after an event. The most critical services needed to assist decision making are those that directly affect viewing and altering event-specific information. Thus, viewing floor plans, entering damage assessments, creating open space reports, and viewing incident commands would be of highest priority. If an emergency or an attack were to occur, it would be crucial to preserve

---

<sup>®</sup> CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

these system functions. Though not quite critical, the case study team flagged two services as important: viewing contact information for maintenance tasks, and finding specialized employees.

The other major functions were all deemed to be non-essential. Though important to the functioning and upkeep of the system, the ability to add this information can be recovered after an attack. Many of the other functions deal with configuring the actual system or its user profiles, which can also be performed after an attack with little loss. Other functions involve making useful but non-critical posts in the form of journal entries or Post-It notes. Still other services support the viewing of non-critical data such as overseas contact information. While all of this functionality is important to the long-term usability of the system, an attack on these services does not threaten the ability of the system to aid decision making during an emergency. If compromised, the information and services would need to be repaired before the system could become fully usable and functional again. However, if the information in the system configuration is kept current, the ability to add new assets and documents during an emergency is secondary to viewing the current state of assets.

#### **4.4.2 Essential Assets**

There were two major assets in the system under study: (1) a server that housed the majority of the system's intellectual assets (i.e., the code that ran the system) and provided remote access and (2) the information inside the central server, including Microsoft IIS configurations, the Sybase database, and the MapGuide Database. These assets were found to be critical in order to make informed decisions.

The user and developer workstations in the system were not considered. No important files or intellectual assets critical to system's mission are housed on these machines. Should they fail, a spare machine could easily act as a replacement, provided the proper software is available. This is not the case with the central server or the information that it contains. An attack on its ability to function, or on its ability to deliver accurate information, will critically impact survivability.

## **4.5 Step 4: Perform Risk Assessment**

In the case study analyses, some risk assessment methodologies were analyzed to determine which were suitable for the elicitation of security requirements. The result of this analysis is presented in this section as an example of techniques that are available, as well as their strengths and weaknesses. These techniques are provided only as examples of previous work, not as recommendations for inclusion in the SQUARE process.

### **4.5.1 Risk Assessment Techniques**

Before the first SQUARE case study, the student team performed a literature review of the available risk assessment techniques. Ideas came from faculty, course work completed by team members at Carnegie Mellon, and Internet and library searches. The search was narrowed down to a list of eight techniques:

- The Government Accountability Office’s (GAO) model [GAO 99]
- National Institute of Standards and Technology (NIST) model [Stoneburner 02]
- NSA’s INFOSEC Assessment Methodology [NSA 05]
- Butler’s Security Attribute Evaluation Method (SAEM) [Butler 02]
- CMU’s “V-RATE” method [Lipson 01]
- Yacov Haimes’s RFRM model [Haimes 04]
- CMU’s Survivable Systems Analysis method [CERT/CC 02]
- Martin Feather’s DDP model [Cornford 04]

After the initial research, the team completed a brief analysis to determine which models would be likely candidates for further consideration. They found that attempts to quantify risks on the basis of dollar value per attack were either too complicated or too involved for the limited time given to the project, and were therefore rejected. The team concluded that qualitative methods would add more value to the short case studies.

The following chart shows which criteria were used to evaluate the different methodologies and how each was scored (using a scale of 1 - 4, with “1” being the highest mark, and “4” being the lowest). Here is a brief explanation of each rating:

1. Very suitable for the requirement
2. Well suited for the requirement
3. Somewhat unsuitable for the requirement
4. Very unsuitable for the requirement

**Table 10: Results of Risk Assessment Literature Review**

	Suitable for small organizations	Suitable for short time frame	Additional data collection required	Suitable for requirements	Average
GAO	2	4	2	2	2.50
<b>NIST</b>	<b>2</b>	<b>2</b>	<b>1</b>	<b>1</b>	<b>1.5</b>
NSA/IAM	3	3	2	2	2.50
SAEM	4	4	4	4	4.00
V-RATE	3	4	4	4	3.75
<b>Haimes</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2.00</b>
SSA	2	2	2	4	2.50
DDP/Feather	3	4	2	4	3.25

As Table 10 illustrates, the student team found that NIST’s SP 800-30, also called the “Risk Management Guide for Information Technology Systems,” and Yacov Haimes’s “Risk Filtering, Ranking, and Management Framework” (RFRM) held the most promise for inclusion in the SQUARE process.

#### **4.5.2 Risk Assessment Field Tests**

The student teams proceeded to conduct an independent field test for each of the two selected methodologies. This section outlines the results of their work.

The RFRM framework contains eight phases, some of which the team found to be out of scope for a basic risk assessment. The team identified and tested two phases of RFRM that it felt were strong candidates for inclusion in SQUARE: Phase III, Bicriteria Filtering and Ranking, and Phase IV, Multicriteria Filtering and Ranking.

NIST’s model for risk assessment is broken into nine steps, each with an output that serves as the input to the next step. Steps 1, 8, and 9 were omitted from the test; Step 1 was completed previously in the SQUARE process, Step 8 deals with control recommendations, which are handled separately in SQUARE, and Step 9, Documentation, was omitted because the team combined these results with the RFRM results. Thus, the steps that were actually included were

- Step 2: Threat Identification
- Step 3: Vulnerability Identification
- Step 4: Control Analysis
- Step 5: Likelihood Determination
- Step 6: Impact Analysis
- Step 7: Risk Determination

The results of the different approaches in these two models produced a list of security risks that are on different levels of abstraction, and thus the two sets of filtered and ranked risk scenarios cannot always be easily compared. In some cases, the results of the two models were in conflict. In other cases, the models produced similar evaluations of risk scenarios. The following chart summarizes a three-tier view of each model's risk assessment results on one of the clients' systems:

*Table 11: Risk Assessment Results*

	<b>NIST</b>	<b>RFRM</b>
Tier 1	<ul style="list-style-type: none"> <li>● Insider or terrorist alters or disables key architecture components.</li> <li>● Insider or terrorist discloses proprietary information.</li> <li>● Terrorist gains unauthorized use of system resources.</li> </ul>	<ul style="list-style-type: none"> <li>● Intruder executes malicious code to gain unauthorized access.</li> <li>● High-level user is recruited for help.</li> <li>● System administrator is recruited for help.</li> <li>● High-level user abuses rights.</li> <li>● System administrator abuses rights.</li> </ul>
Tier 2	<ul style="list-style-type: none"> <li>● Insider installs malicious software (viruses, Trojans, key loggers, etc.).</li> <li>● Insider or natural forces physically destroy system components.</li> <li>● Insider steals system components.</li> </ul>	<ul style="list-style-type: none"> <li>● Intruder sniffs password.</li> <li>● Hardware is damaged by natural disaster or environment.</li> <li>● Intruder socially engineers password.</li> </ul>
Tier 3	<ul style="list-style-type: none"> <li>● Terrorist steals system components.</li> <li>● Terrorist installs malicious software (viruses, Trojans, key loggers, etc.).</li> <li>● Terrorist physically destroys system components.</li> <li>● Insider or terrorist alters or corrupts data.</li> </ul>	<ul style="list-style-type: none"> <li>● Intruder uses abandoned, authenticated browser.</li> <li>● Hardware fails.</li> <li>● Intruders guesses, cracks password.</li> </ul>

The team analyzed the combined results and was able to make the following conclusions:

- Insider threats pose the most important risk to the system.
- Because of weak controls, it is easy for an insider or a passerby to defeat authentication.

Both risk assessment models are concerned with hardware failure or destruction, but they rank the importance differently. Hardware damage is a “Tier 2” risk for both models, but NIST’s output considers deliberate destruction by an insider or terrorist a “Tier 1” risk. Some of the risk scenarios from each model do not map directly to one another. NIST’s output focuses more on an attacker’s motives once inside the system (destroying and corrupting data, disclosing proprietary information, etc.) whereas RFRM’s output deals more with the ability of an attacker to break the frontline defenses of the system.

Every application of the SQUARE Methodology will be unique, and so too will the risk assessment, as it needs to be tailored to meet the context of the system under analysis. What is important is that the results from the risk assessment provide a meaningful way to categorize the likelihood and impact of the major threats to the system.

## **4.6 Step 5: Select Elicitation Techniques**

In order to investigate the applicability of existing requirements elicitation techniques to security requirements, one of the case study teams performed a literature review of the existing techniques. Their results are presented in this section.

### **4.6.1 Literature Review**

The team researched existing, structured elicitation techniques and evaluated them on the following criteria:

- adaptability to security requirements  
The ability of the technique to produce accurate requirements in diverse environments. For example, does the technique apply only to functional requirements?
- CASE tool  
Does the technique have a software tool to complement the process?
- client acceptance  
The likelihood that the client would agree to the elicitation technique in analyzing their requirements. Is the process too invasive in a business environment?
- complexity  
The degree of difficulty in understanding and properly executing the elicitation technique. Can the requirements engineers and stakeholders easily perform the technique correctly once they learn the process?
- graphical output  
The ability of the elicitation technique to produce readily understandable visual artifacts that appeal to the stakeholders.
- implementation duration  
The length of time the requirements engineers and clients need to fully execute the elicitation technique.
- learning curve  
The speed with which the requirements engineers and clients can fully comprehend the elicitation technique.
- maturity  
The time, exposure, and analysis the elicitation technique has experienced in its vetting by the requirements engineering community.
- scalability  
The ability of the elicitation technique to address the requirements of enterprise-level systems, in addition to smaller applications.

Once these criteria were defined, the student team produced the comparison matrix shown in Table 12. The techniques they investigated were misuse cases [Jacobson 92], Soft Systems Methodology (SSM) [Checkland 89], Quality Function Deployment (QFD) [QFD 05], Controlled Requirements Expression (CORE) [Mullery 79], Issue Based Information Systems (IBIS) [Kunz 70], Joint Application Development (JAD) [Wood 89], Feature-Oriented Domain Analysis (FODA) [Kang 90], Critical Discourse Analysis (CDA) [Schiffri 94], and the Accelerated Requirements Method (ARM) [Hubbard 99].

*Table 12: Comparison of Elicitation Techniques*

	Misuse Cases	SSM	QFD	CORE	IBIS	JAD	FODA	CDA	ARM
<b>Adaptability</b>	3	1	3	2	2	3	2	1	2
<b>CASE Tool</b>	1	2	1	1	3	2	1	1	1
<b>Client Acceptance</b>	2	2	2	2	3	2	1	3	3
<b>Complexity</b>	2	2	1	2	3	2	1	1	2
<b>Graphical Output</b>	2	2	1	1	2	1	2	2	3
<b>Implementation Duration</b>	2	2	1	1	2	1	2	2	3
<b>Learning Curve</b>	3	1	2	1	3	2	1	1	1
<b>Maturity</b>	2	3	3	3	2	3	2	2	1
<b>Scalability</b>	1	3	3	3	2	3	2	1	2

Scale: 3 = very good, 2 = fair, 1 = poor.

Based on their comparison, the student team decided to pursue IBIS, JAD, and ARM for further consideration. The result of their experience with each technique is presented in the following section.

## 4.7 Step 6: Elicit Security Requirements

The very first case study team did not use a structured elicitation technique to develop requirements with the stakeholders. In essence, this team utilized the “unstructured interview” method of requirements elicitation. Working with their client, Acme Corporation, they were able to generate the set of requirements shown in Table 13.

*Table 13: Requirements Generated for Acme Using the “Unstructured Interview” Elicitation Technique*

R-01	The system is required to have authentication measures in place at all gateways / entrance points.
R-02	The system is required to have a role-based access control mechanism that governs which system elements (data, functionality, etc.) users can view, modify, and/or interact with.
R-03	It is required that a continuity of operations plan (COOP) be in place to ensure system availability.
R-04	It is required that the AMS’s designated security personnel be able to audit the status and usage of system resources (including security devices).
R-05	The AMS’s designated personnel are required to audit the status of system resources and their usage on a regular basis.
R-06	It is required that the system’s network communications be protected from unauthorized information gathering and/or eavesdropping by encryption and other reasonable techniques.
R-07	It is a requirement that both process-centric and logical means be in place to prevent the installation of any software or device without prior authorization.
R-08	It is required that the AMS’s physical devices be protected against destruction, damage, theft, tampering, or surreptitious replacement (including but not limited to damage due to vandalism, sabotage, terrorism, or acts of God/nature).
R-09	It is required that the AMS’s software components be designed utilizing software security best practices.

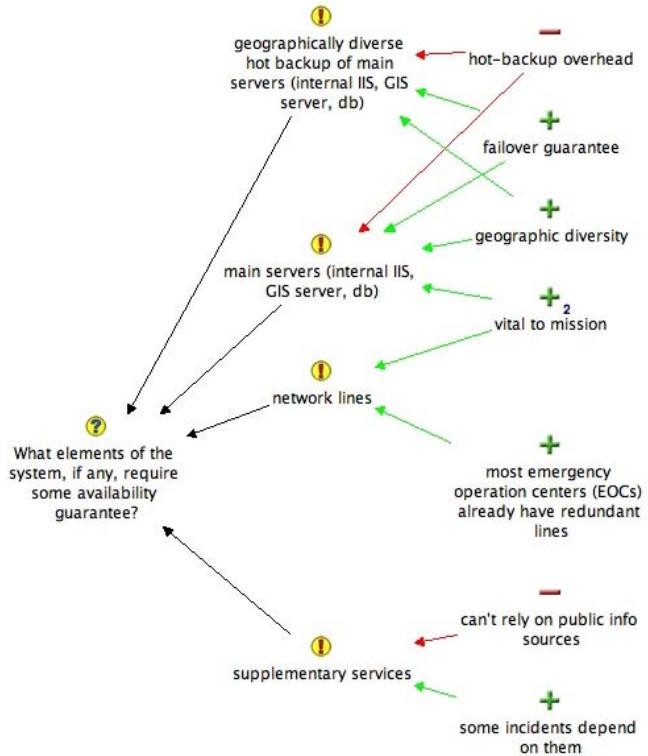
While this technique may have been sufficient, the other case study teams were interested in finding which techniques were more capable of generating more accurate and complete requirements. Another case study team used the results from their elicitation technique literature review and comparison to experiment with Issue Based Information Systems, Joint Application Development, and the Accelerated Requirements Method. The results from their work are presented in this section.

### 4.7.1 IBIS

Issue Based Information System (IBIS) is a technique developed in the 1970s whose goal is to improve the definition, discussion, and resolution of “wicked” problems. That is, the methodology works best with issues that are ill defined or hotly contended among stakeholders.

In IBIS, all problems are decomposed into *issues*, which are phrased in the form of an open question to the stakeholders. For instance, “How should the system guard against insider threats, if at all?” Each issue is then resolved by proposed *positions*, which are resolutions to the issue put forth by the stakeholders. Every position has corresponding *arguments*, which either support or oppose the position.

The requirements engineer is tasked with recording the articulation of issues, positions, and arguments. The results are then presented in the form of an IBIS map. Figure 7 is an example of such a map. The IBIS maps are analyzed by the requirements engineering team and client to elicit the actual security requirements.



*Figure 7: Example IBIS Map Generated with Compendium*

To execute the IBIS technique with Acme Corporation, the team first formulated a set of questions that would likely cover every aspect of security that could affect the system. They tried to come up with any and all questions that would cover confidentiality, availability, and integrity aspects of the system. Many of the questions were based on artifacts collected by the previous SQUARE case study team.

After interviewing the stakeholders with these questions, the team created a set of IBIS maps using the Compendium CASE tool, which is freely available from the Compendium Institute's Web site.<sup>1</sup>

Using the IBIS maps, along with feedback from the stakeholders, the student team was able to generate an initial set of security requirements for the system (see Table 14). Unfortunately, IBIS did not provide a mechanism for translating the maps into requirements, so the requirements were instead based on feedback from the stakeholders during the meetings as well as the team's own recommendations. The full hierarchy of goals to security requirements for Acme can be found in Section 0.

<sup>1</sup> <http://www.compendiuminstitute.org/>

**Table 14: Security Requirements Generated with IBIS**

R-01	The system shall implement authentication via a secure login screen.
R-02	The system shall identify and authenticate all the users who attempt to access the system.
R-03	The server-side components and files contained therein shall have their access restricted to authorized personnel.
R-04	Fault tolerance shall be provided for the system's essential services (IIS server, GIS server, and network lines).
R-05	The system shall maintain data integrity via logged modifications and user access control.
R-06	An access control system shall be configured for information gathering for auditing purposes (access log and application log).
R-07	The system shall recover from any single attack, failure, or accident in less than one minute.
R-08	A backup shall consist of a complete reproduction of every file on the server.
R-09	The system shall be able to provide full functionality from backup.

Given the team's experience with IBIS in eliciting security requirements, they recommended against using it in the future. While IBIS was extremely effective in documenting complex discussions, it did not provide a structured means to generate security requirements based on the resulting maps. They found that for many issues, the stakeholders considered only one position. In turn, they were forced to create an arbitrary number of alternative positions, even if the stakeholders had never considered them. By creating additional positions, the team's work may have benefited the client to some extent by offering new solutions, but they did not feel that this course of action produces the most accurate and complete set of requirements.

#### 4.7.2 ARM

The Accelerated Requirements Method (ARM) is a technique that has been designed to elicit, categorize, and prioritize security requirements. Therefore, ARM stretches over Steps 6, 7, and 8 in the SQUARE process. The case study team spent two weeks completing the ARM process with a real-world client. Their experiences are presented in this section.

At the heart of ARM is the step known as “Brainstorm, Organize, and Name (BON).” In this step, the requirements engineering team and stakeholders meet to develop the initial security requirements. To start the session, the team asked the stakeholders the “focus question,” which was crafted to tie to the previously established goals, objectives, and scope of the project. In this case, they chose the following focus question:

“An important security requirement of the project is \_\_\_\_\_”

Based on their professional experience and security knowledge, the participants were asked to write down seven important security requirements on scratch paper within the time limit of seven minutes.

Afterwards, the team asked each participant to write down their top three security requirements on cards within three minutes. The team then collected the cards and put the candidate security requirements on the wall. The 24 candidate security requirements produced are listed in Table 15.

*Table 15: Initial Set of Security Requirements Produced with ARM*

1	The ability to securely transmit data to remote sources	13	Accountability (who did what, when, how...)
2	The preservation of data integrity	14	Integrity (assurance in data protection and validity)
3	The enforcement and usability of an access control system	15	Indelibility (deletions and retractions are noted/logged)
4	Security must be manageable and not hinder business (where possible)	16	Integrity
5	There must be a strong/reliable authentication process	17	Access control
6	Information must be kept private from the outside world	18	Confidentiality (encryption etc.)
7	Consistent APIs	19	Partitioned data store (public read only and private read/write)
8	Data integrity	20	Selectively secure communication with outside entities)
9	Authentication and access control	21	Represent and support segmented disclosure
10	Strong authentication	22	Role-based restricted views/edit/action access (e.g., summary report info, DC public for particular people)
11	Reduce/eliminate risks of inappropriate use of the system (as defined by policy)	23	Available 24/7 via remote authenticated access and secure
12	Granular access to data for users (operators) and customers	24	Key action audit (e.g., attribution of who pressed the publish button and from where and what changes were made)

The stakeholders then looked through the candidate security requirements generated during the brainstorming session and observed whether any duplicate or inadequate security requirements were produced. They reflected on what they thought were important and defended their own opinions amongst each other. This step provided an opportunity for the stakeholders to share their security concerns about the project. After repeated discussion and debate, they deleted six candidate security requirements that they found to be either redundant or inappropriate. The remaining requirements are listed in Table 16.

**Table 16: Refined Set of Security Requirements Produced with ARM**

3	The enforcement and usability of an access control system	15	Indelibility (deletions and retractions are noted/logged)
4	Security must be manageable and not hinder business (where possible)	18	Confidentiality (encryption etc.)
6	Information must be kept private from the outside world	19	Partitioned data store (public read only and private read/write)
7	Consistent APIs	20	Selectively secure communication with outside entities)
8	Data integrity	21	Represent and support segmented disclosure
10	Strong authentication	22	Role-based restricted views/edit/action access (e.g., summary report info, DC public for particular people)
11	Reduce/eliminate risks of inappropriate behavior	23	Available 24/7 via remote authenticated access and secure
12	Granular access to data for users (operators) and customers	24	Key action audit (e.g., attribution of who pressed the publish button and from where and what changes were made)
13	Accountability (who did what, when, how...)		

As mentioned previously, the remaining steps of ARM cover different steps of the SQUARE process, and are thus described in later sections of this report.

### 4.7.3 JAD

Joint Application Development (JAD) is a very mature, structured technique that was designed for the elicitation of functional requirements of a system. The case study team attempted to utilize JAD with another client project. Their experience is presented here.

The centerpiece of JAD is a structured workshop known as the JAD Session. During the JAD Session, all the stakeholders come together under the facilitation of a requirements engineering team to design a system or piece of software. The phase includes defining work flow, data elements, screens, reports, and open issues. Because JAD is designed mainly for functional requirements, there were some steps that the team found unsuitable for the elicitation of security requirements: the work flow, data elements, screens, and reports steps. Therefore, the team only performed the “open issues” step with the stakeholders.

The stakeholders initially provided sixteen open issues, but only eight of them were security related. Based on collected artifacts, the student team generated eleven open issues:

1. What is your approach to configuration management?
2. What is your approach to defect and issue management?
3. What is the testing methodology to be used in the proposed project?

4. Do you maintain a separate environment for testing, or is testing performed on development servers?
5. How should integrity of the site be protected?
6. How can unauthorized changes to the project affect the mission?
7. What are the best procedures to guarantee these actions are recorded?
8. Does the stipulation that all or most of the code be “open source” present any potential security issues?
9. What do you perceive as the difference between clustering and active/active failover in regards to availability?
10. How will you manage users and authorization?
11. Is 100% uptime necessary for the project?

The team conducted the interview and generated the security requirements based on the stakeholders’ answers. The requirements are listed in Table 17.

**Table 17: Security Requirements Generated with JAD**

R-01	The system shall provide reliable information to the users who have legitimate access to the website.
R-02	The system shall ensure that only authenticated users can access the protected content of the website.
R-03	The system shall protect the privacy of external communications with users.
R-04	The system shall ensure the integrity of content that is provided to the users by using authentication, authorization, and access control.
R-05	The system shall enable version control in both the contents of the website and the development software.
R-06	The system shall enable auditing features that log all content modifications, workflow state transitions, access failures, and authentication attempts.
R-07	The system shall set up clustering to make the service sustainable when disaster occurs.

By not defining the work flow, data elements, screens, and reports of the project, the team found JAD to be very similar to the unstructured interview process. In addition, the JAD session was designed for developing functional requirements, so the team had difficulty in finding a way to discuss non-functional requirements such as security and thus did not recommend JAD for future use in SQUARE.

## 4.8 Step 7: Categorize Requirements

Though none of the student case studies utilized a very structured categorization technique, the Accelerated Requirements Method (ARM) provided a useful means of categorizing security requirements. In this section, the ARM categorization methodology and results are presented.

After the initial requirements were generated by the stakeholders (see Section 4.7.2), the stakeholders were instructed to group the selected security requirements and create a unique name for each group. However, the participants instead engaged in a “spirited” discussion in which they combined the steps of grouping, naming, and categorizing together. In the end, the participants categorized security requirements into six groups, each containing one to four security requirements. Table 18 lists the groups and the requirements contained in each.

*Table 18: Grouped Security Requirements in ARM*

<b>Group A: Confidentiality</b> <ol style="list-style-type: none"><li>1. Information must be kept private from the outside world</li><li>2. Selectively secure communication with outside entities</li></ol>	<b>Group B: Access control</b> <ol style="list-style-type: none"><li>1. Role-based restricted views/edit/action access (e.g., summary report info, public for particular people)</li><li>2. The enforcement and usability of an access control system</li><li>3. Granular access to data for users (operators) and customers</li><li>4. Represent and support segmented disclosure</li></ol>
<b>Group C: Data integrity</b> <ol style="list-style-type: none"><li>1. Partitioned data store (public read only and private read/write)</li><li>2. Indelibility</li></ol>	<b>Group D: Manageability</b> <ol style="list-style-type: none"><li>1. Accountability</li><li>2. Key action audit (e.g., attribution of who pressed the publish button and from where and what changes were made)</li><li>3. Auditing capabilities</li></ol>
<b>Group E: Usability</b> <ol style="list-style-type: none"><li>1. Security must be manageable and not hinder business (where possible)</li><li>2. Available 24/7 via remote authenticated access</li><li>3. Consistent APIs</li><li>4. Reduce/eliminate risks of inadvertent behavior</li></ol>	<b>Group F: Authentication</b> <ol style="list-style-type: none"><li>1. Strong authentication</li></ol>

Two requirements were deleted in this step, and a fourth requirement was added to Group E.

In the future, the requirements engineering team should enforce stricter time limits and attempt to separate the categorization and prioritization processes during this stage.

## 4.9 Step 8: Prioritize Requirements

To prioritize security requirements, one of the case study teams utilized the Analytic Hierarchy Process (AHP) methodology and found it to be very successful both in client acceptance and in its ability to handle security requirements. The technique is briefly presented in this section.

AHP is a technique for decision making in situations where multiple objectives are present. The method calculates the relative value and cost among security requirements. By using AHP, the requirements engineer can also confirm the consistency of the stakeholders' results, which can prevent subjective judgment errors and increase the likelihood that the results are more reliable.

AHP uses a pairwise comparison matrix with all requirements on both axes, as shown in Table 19. Given an arbitrary entry in the matrix, a sub  $i j$ , located in the  $i$ th row and  $j$ th column, the value of a sub  $i j$  indicates how much higher (or lower) the value/cost for requirement  $i$  is than that for requirement  $j$ . In Table 19, the stakeholders only need to fill in the values in the lower left half of the matrix; the remaining values are the reciprocals. The value/cost is measured on an integer scale from 1 to 9, with each number having the interpretation shown in Table 20 and Table 21.

*Table 19: Pairwise Comparison Matrix in AHP*

	A	B	C	D	E	F	G	H	I	J
1		<b>SR-1</b>	<b>SR-2</b>	<b>SR-3</b>	<b>SR-4</b>	<b>SR-5</b>	<b>SR-6</b>	<b>SR-7</b>	<b>SR-8</b>	<b>SR-9</b>
2	<b>SR-1</b>	1	8	1/5	3	1	2	2	3	1
3	<b>SR-2</b>	1/8	1	1/5	1/7	1/7	1/7	1/7	1/9	1/9
4	<b>SR-3</b>	5	5	1	1	2	1	3	1	1
5	<b>SR-4</b>	1/3	7	1	1	1/2	1/2	3	1/2	1
6	<b>SR-5</b>	1	7	1/2	2	1	3	3	1	1/3
7	<b>SR-6</b>	1/2	7	1	2	1/3	1	1/3	1	1
8	<b>SR-7</b>	1/2	7	1/3	1/3	1/3	3	1	3	2
9	<b>SR-8</b>	1/3	9	1	2	1	1	1/3	1	1/6
10	<b>SR-9</b>	1	9	1	1	3	1	1/2	6	1

*Table 20: Interpretation of Values in Matrix*

Intensity of Value	Interpretation of Value
1	Requirements i and j are of equal value.
3	Requirement i has slightly higher value than j.
5	Requirement i has strongly higher value than j.

Intensity of Value	Interpretation of Value
7	Requirement i has very strongly higher value than j.
9	Requirement i has absolutely higher value than j.
2, 4, 6, 8	These are intermediate scales between two adjacent judgments.
Reciprocals	If requirement i has lower value than j.

Table 21: Interpretation of Costs in Matrix

Intensity of Value	Interpretation of Cost
1	Requirements i and j are of equal cost.
3	Requirement i has slightly higher cost than j.
5	Requirement i has strongly higher cost than j.
7	Requirement i has very strongly higher cost than j.
9	Requirement i has absolutely higher cost than j.
2, 4, 6, 8	These are intermediate scales between two adjacent judgments.
Reciprocals	If requirement i has lower cost than j.

After completing the consistency check in AHP, which mathematically estimates the usability of the results, the team was able to produce the cost/value ratio for each requirement. They then created three categories of requirements: high, medium, and low. The results were plotted on a priority graph, which is presented in Figure 8.

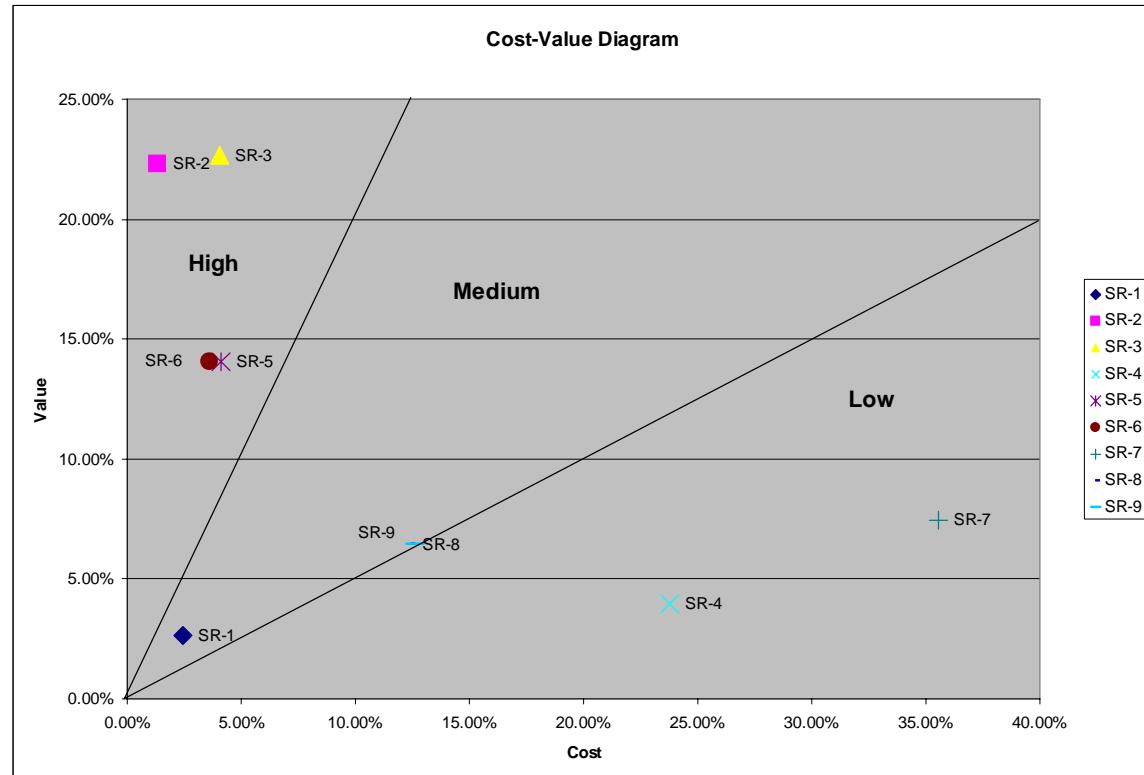


Figure 8: Cost/Value Diagram of Requirements

The stakeholders found AHP valuable not only for its ability to quickly prioritize the security requirements but also because of the internal discussion that is stimulated.

## 4.10 Step 9: Inspect Requirements

The case study teams experimented with different inspection techniques and had different levels of success with each. None of the inspection techniques that were used were sufficiently effective in identifying defects in the security requirements, and the teams do not recommend their use in the future. Instead, the teams recommend that future iterations of SQUARE experiment with the Fagan inspection technique, which is a highly structured and proven technique for requirements inspection. For reference purposes, the inspection techniques and their results are presented in this section.

The first case study team utilized a *peer review log* to find defects in their security requirements. The methodology assigns each team member inspection responsibilities and develops a log that ranks problems according to their severity. The peer review log is a spreadsheet that provides serial number, date, origin, defect type, defect description, defect severity, owner, reviewer, and status of inspection. The following is a blank snapshot of the log:

Table 22: Peer Review Log Format

SNO	DATE	ORIGIN	DEFECT TYPE	DESCRIPTION	SEVERITY	OWNER	REVIEWER	STATUS

The following is a description of the elements that make up the peer review log:

- **SNO**  
The inspection report begins with this serial number of the inspection log. The format used for serial numbers is SNO-xx.
- **Date**  
The format used for dates is mm/dd/yyyy.
- **Origin**  
The section where the defect occurs. The format used for this field is Doc-xx, Page xx. For instance, “Doc-02: Security Requirements Document, Page 12.”
- **Defect Type**  
One of the following: missing content, unclear/ambiguous, lack of understanding of requirements, oversight, repeated occurrence of an error, undefined acronyms, or abbreviations.
- **Description**  
A few sentences illustrating the problem.
- **Severity**
  1. High: Could jeopardize the project success.

2. Moderate: Problem that requires correction before proceeding.
3. Low: Cosmetic or style problem.

- **Owner**

The person identified as the original author.

- **Reviewer**

The person identified as the inspector.

- **Status**

Either: Open (1) or Closed (0).

The following case study teams used a checklist method of requirements inspection. The idea behind this technique is to provide the stakeholders with a set of questions that encourage them to review the security requirements and determine the quality of each. Table 23 shows the checklist that the team provided.

*Table 23: Inspection Checklist*

<b>Organization:</b>
<ul style="list-style-type: none"> <li>• Are all requirements written at a consistent and appropriate level of detail?</li> <li>• Is the implementation priority of each requirement included?</li> <li>• Do the requirements include all of the known customer and system needs?</li> <li>• Is any necessary information missing from a requirement?</li> <li>• Are there areas not addressed in the document that need to be?</li> <li>• Is the document well organized?</li> <li>• Are there requirements that contain an unnecessary level of design detail?</li> </ul>
<b>Correctness:</b>
<ul style="list-style-type: none"> <li>• Do any requirements conflict with or duplicate other requirements?</li> <li>• Is each requirement written in clear, concise, unambiguous language?</li> <li>• Is each requirement verifiable by testing, demonstration, review, or analysis?</li> <li>• Is each requirement in scope for the project?</li> <li>• Is each requirement free from content and grammatical errors?</li> <li>• Can all of the requirements be implemented within known constraints?</li> </ul>
<b>Traceability:</b>
<ul style="list-style-type: none"> <li>• Is each requirement uniquely and correctly identified?</li> </ul>
<b>Special Issues:</b>
<ul style="list-style-type: none"> <li>• Are all requirements actually requirements, not design or implementation solutions?</li> </ul>

Unfortunately, the stakeholders found the checklist to be overwhelming and time consuming. In addition, there is no guarantee that the questions on the checklist will identify all defects in the security requirements. Because of these problems with the method, the team does not recommend that checklists should be used for requirements inspection.

---

## 5 Future Research Plans

NSS's final goal for SQUARE is full industry adoption and integration, but currently it is still being improved. In addition to performing new case studies with the process, NSS is currently developing a Web-based CASE tool to support the methodology. The tool will assist the requirements engineering team in each step of the SQUARE process by automating documentation and streamlining communication with stakeholders. A prototype of this tool will be available in the near future.

A recommendation from the latest case study is to combine the elicitation and categorization strengths of ARM with the prioritization efficiency of AHP. NSS would like to experiment with the combination of these techniques on a new client organization.

Also, there are two steps of the SQUARE process that NSS has immediate plans to investigate further. The first is the existence and applicability of structured categorization methods that could work when applied to security requirements. In the previous case studies, categorization was done in an unstructured manner with the stakeholders. Secondly, NSS would like to utilize the Fagan inspection process in Step 9 of the SQUARE process. Both of these steps will undergo experimentation during the next SQUARE case study.



---

# Appendix

## Definitions from Initial Case Study

*Table 24: Terms and Definitions from Initial Case Study*

<b>access control</b>	Access control ensures that resources are only granted to those users who are entitled to them.
<b>access control list</b>	A table that tells a computer operating system which access rights or explicit denials each user has to a particular system object, such as a file directory or individual file [TechTarget 05].
<b>antivirus software</b>	A program that searches hard drives and floppy disks for any known or potential viruses [TechTarget 05].
<b>artifact</b>	The remnants of an intruder attack or incident activity. These could be software used by intruder(s), a collection of tools, malicious code, logs, files, output from tools, or the status of a system after an attack or intrusion [West-Brown 03].
<b>asset</b>	A critical valuable that a company owns and wants to secure.
<b>attack</b>	An action conducted by an adversary, the attacker, on a potential victim. A set of events that an observer believes to have information assurance consequences on some entity, the target of the attack [Ellison 03].
<b>auditing</b>	The information gathering and analysis of assets to ensure such things as policy compliance and security from vulnerabilities [SANS 05].
<b>authentication</b>	The process of determining whether someone or something is, in fact, who or what it is declared to be [TechTarget 05].
<b>availability</b>	The property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system; i.e., a system is available if it provides services according to the system design whenever users request them [Allen 99].
<b>back door</b>	An element in a system that allows access by bypassing access controls [Howard 97].
<b>breach</b>	Any intentional event in which an intruder gains access that compromises the confidentiality, integrity, or availability of computers, networks, or the data residing on them [CERT/CC 05].
<b>brute force</b>	A cryptanalysis technique or other kind of attack method involving an exhaustive procedure that tries all possibilities, one by one [SANS 05].
<b>buffer overflow</b>	A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information—which has to go somewhere—can overflow into adjacent buffers, corrupting or overwriting the valid data held in them [SANS 05].
<b>cache cramming</b>	The technique of tricking a browser to run cached Java code from the local disk instead of the Internet zone, so it runs with less restrictive permissions [SANS 05].
<b>cache poisoning</b>	Malicious or misleading data from a remote name server is saved [cached] by another name server. Typically used with Domain Name System (DNS) cache poisoning attacks [SANS 05].

<b>confidentiality</b>	The property that information is not made available or disclosed to unauthorized individuals, entities, or processes (i.e., to any unauthorized system entity) [SANS 05].
<b>control</b>	An action, device, procedure, or technique that removes or reduces a vulnerability
<b>corruption</b>	A threat action that undesirably alters system operation by adversely modifying system functions or data [SANS 05].
<b>cracker</b>	Someone who breaks into someone else's computer system, often on a network; bypasses passwords or licenses in computer programs; or in other ways intentionally breaches computer security [TechTarget 05].
<b>denial-of-service (DoS) attack</b>	A form of attacking another computer or company by sending millions of requests every second, causing the network to slow down, cause errors, or shut down.
<b>disaster recovery plan</b>	A disaster recovery plan (DRP)—sometimes referred to as a business continuity plan (BCP) or business process contingency plan (BPCP)—describes how an organization is to deal with potential disasters [TechTarget 05].
<b>disclosure</b>	The dissemination of information to anyone who is not authorized to access that information [Alberts 03].
<b>disgruntled employee</b>	A person in an organization who deliberately abuses or misuses computer systems and their information [Alberts 03].
<b>downtime</b>	The amount of time a system is down in a given period. This will include crashes and system problems as well as scheduled maintenance work [RUsecure 05].
<b>disruption</b>	A circumstance or event that interrupts or prevents the correct operation of system services and functions [Alberts 03].
<b>encryption</b>	Cryptographic transformation of data (called "plaintext") into a form (called "cipher text") that conceals the data's original meaning to prevent it from being known or used [SANS 05].
<b>espionage</b>	The act or practice of spying or of using spies to obtain secret information about another government or a business competitor [Dictionary.com 05].
<b>essential services</b>	Services to users of a system that must be provided even in the presence of intrusion, failure, or accident [Ellison 97].
<b>exposure</b>	Same as <b>disclosure</b> .
<b>fabrication</b>	Same as <b>masquerade</b> .
<b>fault line attacks</b>	Fault line attacks use weaknesses between interfaces of systems to exploit gaps in coverage [SANS 05].
<b>fault tolerance</b>	Describes a computer system or component designed so that, in the event that a component fails, a backup component or procedure can immediately take its place with no loss of service. Fault tolerance can be provided with software, or embedded in hardware, or provided by some combination [TechTarget 05].
<b>firewall</b>	A system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both [Webopedia 05].
<b>hacker</b>	An individual who breaks into computers primarily for the challenge and status of obtaining access [Howard 97].
<b>honey pot</b>	Programs that simulate one or more network services designated on a computer's ports. An attacker assumes that vulnerable services that can be used to break into the machine are being run. A honey pot can be used to log access attempts to those ports, including the attacker's keystrokes. This can provide advanced warning of a more concerted attack [SANS 05].
<b>HTTP header manipulation</b>	HTTP requests and responses send information in the HTTP headers. HTTP headers are a series of lines containing a name/value pair used to pass information such as the host, referrer, user agent, etc. HTTP headers can be manipulated to cause SQL injection or cross-site scripting errors.

<b>impact</b>	The negative effect of an attack on a victim system by an attacker [Allen 99].
<b>incident</b>	An adverse network event in an information system or network or the threat of the occurrence of such an event [SANS 05].
<b>incident handling</b>	An action plan for dealing with intrusions, cyber theft, denial of service, fire, floods, and other security-related events [SANS 05].
<b>insider threat</b>	The threat that authorized personnel of an organization will act counter to the organization's security and interest, especially for the purposes of sabotage and espionage [NIPC 02].
<b>integrity</b>	For systems, the quality that a system has when it can perform its intended function in a unimpaired manner, free from deliberate or inadvertent unauthorized manipulation. For data, the property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner [Allen 99].
<b>interception</b>	Access to an asset gained by an unauthorized party [Pfleeger 03].
<b>interruption</b>	An event that causes an asset of a system to be destroyed or become unavailable or unusable [Howard 97].
<b>intrusion</b>	An attack on a network for the purpose of gaining access to or destroying privileged information or disrupting services to legitimate users [Ellison 03].
<b>intrusion detection system</b>	A combination of hardware and software that monitors and collects system and network information and analyzes it to determine if an attack or an intrusion has occurred. Some ID systems can automatically respond to an intrusion [Allen 99].
<b>intrusion prevention system</b>	A system used to actively drop packets of data or disconnect connections that contain unauthorized data. Intrusion prevention technology is also commonly an extension of intrusion detection technology [Wiki 05].
<b>liability</b>	The responsibility of someone for damage or loss [West-Brown 03].
<b>luring attack</b>	A type of elevation of privilege attack where the attacker "lures" a more highly privileged component to do something on his or her behalf. The most straightforward technique is to convince the target to run the attacker's code in a more privileged security context [Brown 05].
<b>malware</b>	Programming or files that are developed for the purpose of doing harm. Thus, malware includes computer viruses, worms, and Trojan horses [Webopedia 05].
<b>man-in-the-middle attack</b>	An attack in which the attacker is able to read, and possibly modify at will, messages between two parties without letting either party know that they have been attacked. The attacker must be able to observe and intercept messages going between the two victims [Farlex 05].
<b>masquerade</b>	Aims to fool other machines on the network into accepting the imposter as an original, either to lure the other machines into sending it data or to allow it to alter data [Howard 98].
<b>modification</b>	Situation in which an unauthorized party not only gains access to, but tampers with an asset [Howard 97].
<b>non-essential services</b>	Services to users of a system that can be temporarily suspended to permit delivery of essential services while the system is dealing with intrusions and compromises [Ellison 97].
<b>non-repudiation</b>	The goal of non-repudiation is to prove that a message has been sent and received [SSI 05].
<b>patch</b>	A small update released by a software manufacturer to fix bugs in an existing program [SANS 05].
<b>patching</b>	The process of updating software to a new version that fixes bugs in a previous version [SANS 05].
<b>penetration</b>	Intrusion, trespassing, or unauthorized entry into a system [RUsecure 05].

<b>penetration testing</b>	The execution of a testing plan, the sole purpose of which is to attempt to hack into a system using known tools and techniques [RUsure 05].
<b>physical security</b>	Security measures taken to protect systems, buildings, and related supporting infrastructure against threats associated with their physical environment [Guttman 95].
<b>port scanning</b>	The act of systematically scanning a computer's ports [Webopedia 05].
<b>privacy</b>	The quality or condition of being secluded from the presence or view of others [Dictionary.com 05].
<b>procedure</b>	The implementation of a policy in the forms of workflows, orders, or mechanisms [West-Brown 03].
<b>recognition</b>	The capability of a system to recognize attacks or the probing that precedes attacks [Ellison 03].
<b>recovery</b>	A system's ability to restore services after an intrusion has occurred. Recovery also contributes to a system's ability to maintain essential services during intrusion [Ellison 03].
<b>replay attack</b>	The interception of communications, such as an authentication communication, and subsequent impersonation of the sender by retransmitting the intercepted communication [FFIEC 04].
<b>resilience</b>	The ability of a computer or system to both withstand a range of load fluctuations and also remain stable under continuous and/or adverse conditions [RUsure 05].
<b>resistance</b>	Capability of a system to resist attacks [Ellison 03].
<b>risk</b>	The product of the level of threat with the level of vulnerability. It establishes the likelihood of a successful attack [SANS 05].
<b>risk assessment</b>	The process by which risks are identified and the impact of those risks determined [SANS 05].
<b>security policy</b>	A policy that addresses security issues [West-Brown 03].
<b>script kiddies</b>	The more immature but unfortunately often just as dangerous exploiter of security lapses on the Internet. The typical script kiddy uses existing and frequently well known and easy-to-find techniques and programs or scripts to search for and exploit weaknesses in other computers on the Internet—often randomly and with little regard or perhaps even understanding of the potentially harmful consequences [TechTarget 05].
<b>spoof</b>	The term is used to describe a variety of ways in which hardware and software can be fooled. IP spoofing, for example, involves trickery that makes a message appear as if it came from an authorized IP address [Webopedia 04].
<b>SQL injection</b>	A type of input validation attack specific to database-driven applications where SQL code is inserted into application queries to manipulate the database [SANS 05].
<b>stakeholder</b>	Anyone who is a direct user, indirect user, manager of users, senior manager, operations staff member, support (help desk) staff member, developer working on other systems that integrate or interact with the one under development, or maintenance professionals potentially affected by the development and/or deployment of a software project [Ambler 04].
<b>stealthing</b>	A term that refers to approaches used by malicious code to conceal its presence on an infected system [SANS 05].
<b>survivability</b>	The capability of a system to complete its mission in a timely manner, even if significant portions are compromised by attack or accident. The system should provide essential services in the presence of successful intrusion and recover compromised services in a timely manner after intrusion occurs [Mead 03].
<b>target</b>	The object of an attack, especially host, computer, network, system, site, person, organization, nation, company, government, or other group [Allen 99].
<b>threat</b>	A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm [SANS 05].

<b>threat assessment</b>	The identification of the types of threats that an organization might be exposed to [SANS 05].
<b>threat model</b>	Used to describe a given threat and the harm it could do to a system if it has a vulnerability [SANS 05].
<b>toolkits</b>	A collection of tools with related purposes or functions, e.g., antivirus toolkit, disk toolkit [RUsecure 05].
<b>Trojan</b>	A program in which malicious or harmful code is contained inside apparently harmless programming or data in such a way that it can get control and do its chosen form of damage, such as ruining the file allocation table on a hard disk [TechTarget 05].
<b>trust</b>	Determines which permissions other systems or users have and what actions they can perform on remote machines [SANS 05].
<b>uptime</b>	Same as <b>availability</b> .
<b>victim</b>	That which is the target of an attack. An entity may be a victim of either a successful or unsuccessful attack [SANS 05].
<b>virus</b>	A hidden, self-replicating section of computer software, usually malicious logic, that propagates by infecting—i.e., inserting a copy of itself into and becoming part of—another program. A virus cannot run by itself; it requires that its host program be run to make it active [SANS 05].
<b>vulnerability</b>	A condition or weakness in (or absence of) security procedures, technical controls, physical controls, or other controls that could be exploited by a threat [Guttmann 95].
<b>worm</b>	A self-replicating virus that does not alter files but resides in active memory and duplicates itself. Worms use parts of an operating system that are automatic and usually invisible to the user. It is common for worms to be noticed only when their uncontrolled replication consumes system resources, slowing or halting other tasks [TechTarget 05].

# Goal Hierarchy for Acme Corporation

*Table 25: Goal Hierarchy for Acme Corporation*

<b>Business Goal</b>		
The system allows the client to make informed decisions based on which assets are available.		
<b>Security Goals</b>		
G-01 Management shall exercise effective control over the system's configuration and usage. G-02 The confidentiality, accuracy, and integrity of the system's data shall be maintained. G-03 The system shall be available for use when needed.		
Security Requirements		Refers to goal(s):
R-01 The system is required to have strong authentication measures in place at all system gateways/entrance points.		G-01, 02
R-02 The system is required to have sufficient process-centric and logical means to govern which system elements (data, functionality, etc.) users can view, modify, and/or interact with.		G-01, 02
R-03 It is required that a continuity of operations plan (COOP) be in place to ensure appropriate system availability.		G-03
R-04 It is required that the system's designated security personnel be able to audit the status and usage of system resources (including security devices).		G-01
R-05 The system's designated personnel are required to audit the status of system resources and their usage on a regular basis.		G-01
R-06 It is required that the system's network communications be protected from unauthorized information gathering and/or eavesdropping by encryption and other reasonable techniques.		G-01, 02
R-07 It is a requirement that both process-centric and logical means be in place to prevent the installation of any software or device without prior authorization.		G-01
R-08 It is required that the system's physical devices be protected against destruction, damage, theft, tampering or surreptitious replacement (including but not limited to damage due to vandalism, sabotage, terrorism, or acts of God/nature).		G-03
R-09 It is required that the system's software components be designed utilizing software security best practices.		G-02, 03

---

# References

URLs are valid as of the publication date of this document.

- [Alberts 03]** Alberts, Christopher & Dorofee, Audrey. *OCTAVE Threat Profiles*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.  
<http://www.cert.org/archive/pdf/OCTAVEthreatProfiles.pdf>.
- [Allen 99]** Allen, J.; Christie, A.; Fithen, W.; McHugh, J.; Pickel, J.; & Stoner, E. *State of the Practice of Intrusion Detection Technologies* (CMU/SEI-99-TR-028, ADA375846). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999.  
<http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html>.
- [Ambler 04]** Ambler, Scott W. *Active Stakeholder Participation*. Evergreen, Colorado, Ronin International, Inc., 2004.  
<http://www.agilemodeling.com/essays/activeStakeholderParticipation.htm>.
- [Boehm 88]** Boehm, B. W. & Papaccio, P. N. “Understanding and Controlling Software Costs.” *IEEE Transactions on Software Engineering SE-4*, 10 (October 1988): 1462-77.
- [Boehm 01]** Boehm, Barry; Grünbacher, Paul; & Briggs, Robert O. “Developing Groupware for Requirements Negotiation: Lessons Learned.” *IEEE Software 18*, 2 (May/June 2001).
- [Brown 05]** Brown, Keith. “Item 7: What is a Luring Attack?” *The .NET Developer’s Guide to Windows Security*. Boston, MA: Addison-Wesley, 2005.
- [Butler 02]** Butler, Shawn. “Security Attribute Evaluation Method: A Cost-Benefit Approach,” 232-240. *Proceedings of the 24th International Conference on Software Engineering*. Orlando, FL, May 19-25, 2002. New York NY: ACM Press, 2002.
- [CERT/CC 02]** CERT Coordination Center. *Survivable Systems Analysis Method*. CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, 2002.  
<http://www.cert.org/archive/html/analysis-method.html>.

- [CERT/CC 05]** CERT Coordination Center. *Responding to Intrusions*. <http://www.cert.org/security-improvement/modules/m06.html> (2005).
- [Checkland 89]** Checkland, Peter. *Soft Systems Methodology. Rational Analysis for a Problematic World*. New York, NY: John Wiley & Sons, 1989.
- [Chen 04]** Chen, P.; Dean, M.; Ojoko-Adams, D.; Osman, H.; Lopez, L.; & Xie, N. *Systems Quality Requirements Engineering (SQUARE) Methodology: Case Study on Asset Management System* (CMU/SEI-2004-SR-015). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. <http://www.sei.cmu.edu/publications/documents/04.reports/04sr015.html>.
- [Cornford 04]** Cornford, Steven L.; Feather, Martin S.; & Hicks, Kenneth A. *DDP – A Tool for Life-Cycle Risk Management*. <http://ddptool.jpl.nasa.gov/docs/f344d-slc.pdf> (2004).
- [Davis 03]** Davis, Alan M. “The Art of Requirements Triage.” *Computer* 36, 3 (March 2003): 42-49.
- [Dictionary.com 05]** Dictionary.com. <http://dictionary.reference.com/> (2005).
- [Ellison 97]** Ellison, B.; Fisher, D. A.; Linger, R. C.; Lipson, H. F.; Longstaff, T.; & Mead, N. R. *Survivable Network Systems: An Emerging Discipline* (CMU/SEI-97-TR-013, ADA341963). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. <http://www.sei.cmu.edu/publications/documents/97.reports/97tr013/97tr013abstract.html>.
- [Ellison 03]** Ellison, R. & Moore, A. *Trustworthy Refinement Through Intrusion-Aware Design* (CMU/SEI-2003-TR-002, ADA414865). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr002.html>.
- [Fagan 86]** Fagan, Michael E. “Advances in Software Inspections.” *IEEE Transactions on Software Engineering SE-12*, 7 (July 1986).
- [Farlex 05]** Farlex, Inc. “Man in the middle attack.” *TheFreeDictionary.com*. <http://encyclopedia.thefreedictionary.com/man%20in%20the%20middle%20attack> (2005).

- [FFIEC 04]** FFIEC. “Booklet: Information Security Section: Appendix B: Glossary.” Washington, D.C., Federal Financial Institutions Examination Council, 2004.  
[http://www.ffiec.gov/ffiecinfobase/booklets/information\\_security/08\\_glossary.html](http://www.ffiec.gov/ffiecinfobase/booklets/information_security/08_glossary.html).
- [GAO 99]** General Accounting Office. “Information Security Risk Assessment: Practices of Leading Organizations, A Supplement to GAO’s May 1998 Executive Guide on Information Security Management.” Washington, DC: U.S. General Accounting Office, 1999.
- [Gordon 05]** Gordon, D.; Stehney, T.; Wattas, N.; & Yu, E. *Quality Requirements Engineering (SQUARE): Case Study on Asset Management System, Phase II* (CMU/SEI-2005-SR-005). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.  
<http://www.sei.cmu.edu/publications/documents/05.reports/05sr005.html>.
- [Guttman 95]** Guttman, Barbara; Roback, Edward. *An Introduction to Computer Security*. Gaithersburg, MD: U.S. Department of Commerce, Technology Administration, National Institute of Standards and Technology, 1995. <http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf>.
- [Haimes 04]** Haimes, Yacov Y. *Risk Modeling, Assessment, and Management*, 2nd ed. Hoboken, NJ: John Wiley and Sons, Inc., 2004.
- [Howard 97]** Howard, John D. *An Analysis of Security Incidents on the Internet 1989-1995*. Pittsburgh, Pa., Carnegie Mellon University, Software Engineering Institute, 1997.  
<http://www.cert.org/research/JHThesis/Start.html>.
- [Howard 98]** Howard, John; Longstaff, Thomas. *A Common Language for Computer Security Incidents*. Albuquerque, N.M., Sandia National Laboratories, 1998.  
[http://www.cert.org/research/taxonomy\\_988667.pdf](http://www.cert.org/research/taxonomy_988667.pdf).
- [Hubbard 99]** Hubbard, R. “Design, Implementation, and Evaluation of a Process to Structure the Collection of Software Project Requirements.” PhD diss., Colorado Technical University, 1999.
- [IEEE 90]** IEEE. *IEEE Standard Glossary of Software Engineering Terminology* (IEEE Std 610.12-1990). Los Alamitos, CA: IEEE Computer Society Press, 1990.
- [IEEE 05]** IEEE. *Guide to the Software Engineering Body of Knowledge*.  
<http://www.swebok.org/> (2005).

- [Jacobson 92]** Jacobson, Ivar. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Boston, MA: Addison-Wesley, 1992.
- [Jones 86]** Jones, Capers, ed. *Tutorial: Programming Productivity: Issues for the Eighties, 2nd Ed.* Los Angeles, CA: IEEE Computer Society Press, 1986.
- [Kang 90]** Kang, K.; Cohen, S.; Hess, J.; Novak, W.; & Peterson, A. *Feature-Oriented Domain Analysis (FODA) Feasibility Study* (CMU/SEI-90-TR-021, ADA235785). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990.  
<http://www.sei.cmu.edu/publications/documents/90.reports/90.tr.021.html>.
- [Karlsson 97]** Karlsson, J. & Ryan K. “A Cost-Value Approach for Prioritizing Requirements.” *IEEE Software* 14, 5 (Sept./Oct. 1997): 67-74.
- [Kunz 70]** Kunz, Werner & Rittel, Horst. “Issues as Elements of Information Systems.” <http://www-iurd.ced.berkeley.edu/pub/WP-131.pdf> (1970).
- [Lipson 01]** Lipson, Howard F.; Mead, Nancy R.; & Moore, Andrew P. *A Risk-Management Approach to the Design of Survivable COTS-Based Systems*. <http://www.cert.org/research/isw/isw2001/papers/Lipson-29-08-a.pdf> (2001).
- [McConnell 01]** McConnell, Steve. “From the Editor - An Ounce of Prevention.” *IEEE Software* 18, 3 (May 2001): 5-7.
- [Mead 03]** Mead, Nancy. *Requirements Engineering for Survivable Systems* (CMU/SEI-2003-TN-013, ADA418410). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.  
<http://www.sei.cmu.edu/publications/documents/03.reports/03tn013.html>.
- [Moffett 04]** Moffett, Jonathan D.; Haley, Charles B.; & Nuseibeh, Bashar. *Core Security Requirements Artefacts* (Technical Report 2004/23). Milton Keynes, UK: Department of Computing, The Open University, June 2004.
- [Mullery 79]** Mullery, G. P. “CORE: A Method for Controlled Requirements Specification.” *Proceedings of the 4th International Conference on Software Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1979.

- [NIPC 02]** NIPC Special Technologies and Applications Unit (STAU). “Insiders and Information Technology.” Washington, D.C.: Department of Homeland Security, Information Analysis Infrastructure Protection, 2002. <http://www.hpcc-usa.org/pics/02-pres/wright.ppt>.
- [NIST 02]** National Institute of Standards and Technology. “Software Errors Cost U.S. Economy \$59.5 Billion Annually” (NIST 2002-10). [http://www.nist.gov/public\\_affairs/releases/n02-10.htm](http://www.nist.gov/public_affairs/releases/n02-10.htm) (2002).
- [NSA 05]** National Security Agency. *INFOSEC Assessment Methodology*. <http://www.iatrp.com/iam.cfm> (2005).
- [Pfleeger 03]** Pfleeger, Charles P. & Pfleeger, Shari Lawrence. *Security in Computing*, 3rd ed. Upper Saddle River, NJ: Prentice Hall PTR, 2003.
- [QFD 05]** QFD Institute. *Frequently Asked Questions About QFD*. [http://www.qfdi.org/what\\_is\\_qfd/faqs\\_about\\_qfd.htm](http://www.qfdi.org/what_is_qfd/faqs_about_qfd.htm) (2005).
- [RUsecure 05]** RUsecure. *The Information Security Glossary*. <http://www.yourwindow.to/information-security> (2005).
- [Saaty 80]** Saaty, T. L. *The Analytic Hierarchy Process*. New York, NY: McGraw-Hill, 1980.
- [Sans 05]** The SANS Institute. *SANS Glossary of Terms Used in Security and Intrusion Detection*. <http://www.sans.org/resources/glossary.php> (2005).
- [Schiffrin 94]** Schiffrin, D. *Approaches to Discourse*. Oxford, England: Blackwell, 1994.
- [Soo Hoo 01]** Soo Hoo, Kevin; Sudbury, Andrew W.; & Jaquith, Andrew R. “Tangible ROI through Secure Software Engineering.” *Secure Business Quarterly* 1, 2 (2001).
- [SSI 05]** Service Strategies Inc. “Nonrepudiation.” *Glossary of Messaging & Security Terms*. <http://www.ssimal.com/Glossary.htm#N> (2005).
- [Stoneburner 02]** Stoneburner, Gary; Goguen, Alice; & Feringa, Alexis. *Risk Management Guide for Information Technology Systems* (Special Publication 800-30). Gaithersburg, MD: National Institute of Standards and Technology, 2002. <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>.
- [TechTarget 05]** TechTarget. Whatis?com: The Leading IT Encyclopedia and Learning Center. <http://whatis.techtarget.com> (2005).

- [Watro 01]** Watro, R. J. & Shirley, R. W. "Mapping Mission-Level Availability Requirements to System Architectures and Policy Abstractions," 189-199. *Proceedings of DARPA Information Survivability Conference & Exposition II, 2001* (Vol. 1). June 12-14, 2001. Los Alamitos, CA: IEEE Computer Society Press, 2001.
- [Webopedia 05]** Webopedia. <http://www.webopedia.com/> (2005).
- [West-Brown 03]** West-Brown, M.; Stikvoort, D.; Kossakowski, K.; Killcrece, G.; Ruefle, R.; & Zajicek, M. *Handbook for Computer Security Incident Response Teams (CSIRTs)* (CMU/SEI-2003-HB-002). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03hb002.html>.
- [Wiegers 01]** Wiegers, Karl E. *Inspecting Requirements* (column). StickyMinds, July 30, 2001. <http://www.stickyminds.com>.
- [Wiegers 02]** Wiegers, Karl E. *Peer Reviews in Software: A Practical Guide*. Boston, MA: Addison-Wesley, 2002.
- [Wiki 05]** Wikipedia: The Free Encyclopedia. <http://www.wikipedia.org/> (2005).
- [Wood 89]** Wood, Jane & Silver, Denise. *Joint Application Design: How to Design Quality Systems in 40% Less Time*. New York, NY: John Wiley & Sons, 1989.

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved OMB No. 0704-0188</i>
<p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</p>			
1. AGENCY USE ONLY  (Leave Blank)	2. REPORT DATE  November 2005	3. REPORT TYPE AND DATES COVERED  Final	
4. TITLE AND SUBTITLE  Security Quality Requirements Engineering (SQUARE) Methodology	5. FUNDING NUMBERS  FA8721-05-C-0003		
6. AUTHOR(S)  Nancy R. Mead, Eric D. Hough, Theodore R. Stehney II			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	8. PERFORMING ORGANIZATION REPORT NUMBER  CMU/SEI-2005-TR-009		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	10. SPONSORING/MONITORING AGENCY REPORT NUMBER  ESC-TR-2005-009		
11. SUPPLEMENTARY NOTES			
12A DISTRIBUTION/AVAILABILITY STATEMENT  Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE  81	
<p>13. ABSTRACT (MAXIMUM 200 WORDS)</p> <p>Requirements engineering, a vital component in successful project development, often does not include sufficient attention to security concerns. Studies show that up-front attention to security can save the economy billions of dollars, yet security concerns are often treated as an afterthought to functional requirements. Industry can thus benefit from a model to examine security requirements in the development stages of the production life cycle.</p> <p>This report presents the Security Quality Requirements Engineering (SQUARE) Methodology for eliciting and prioritizing security requirements in software development projects, which was developed by the Software Engineering Institute's Networked Systems Survivability (NSS) Program. The methodology's steps are explained and results from its application in recent case studies are examined. The NSS Program continues to develop SQUARE, which has proven effective in helping organizations understand their security posture and produce products with verifiable security requirements.</p>			
14. SUBJECT TERMS  information security improvement, misuse cases, requirements engineering, system survivability		15. NUMBER OF PAGES  80	
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT  Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE  Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT  Unclassified	20. LIMITATION OF ABSTRACT  UL